IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification

Sponsor

Simulation Interoperability Standards Committee
of the
IEEE Computer Society

Approved 21 September 2000

IEEE-SA Standards Board

Abstract: The High Level Architecture (HLA)—Object Model Template (OMT) specification defines the format and syntax (but not content) of HLA object models. Simulations are abstractions of the real world, and no one simulation can solve all of the functional needs for the modeling and simulation community. It is anticipated that advances in technology will allow for new and different modeling and simulation (M&S) implementations within the framework of the HLA. The standards contained in this architecture are interrelated and need to be considered as a product set, as a change in one is likely to have an impact on the others. As such, the HLA is an integrated approach that has been developed to provide a common architecture for simulation.
Keywords: architecture, class attribute, data distribution management, federate, federation, federation execution, federation object model, framework, high level architecture, instance attribute, instance attribute ownership, interaction class, joined federate, object class, object model template, rules, runtime infrastructure, simulation object model, time-constrained, time-regulating

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-2623-3 SH94884
PDF: ISBN 0-7381-2624-1 SS94884

Introduction

(This introduction is not part of IEEE Std 1516.2-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification.)

This document has been developed to record an international standard for the HLA. It serves as one of three related standards for the HLA. It defines the format and syntax for recording information in HLA object models.

Participants

At the time this standard was completed, the working group on HLA, sponsored by the Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society, had the following membership:

Susan F. Symington, *Chair*
Katherine L. Morse, *Vice Chair*
Mikel Petty, *Secretary*

Stephen Bachinsky
Joe Batman
Brian Beebe
Keith Briggs
Jim Calvin
Richard M. Fujimoto
Deborah Fullford
Allison Griffin
James H. Hammond
James Ivers
Andreas Kemkes
John F. Kramer
Fredrick S. Kuhl
Gary M. Lightner
Reed Little
Rodney Long
Margaret Loper
Robert R. Lutz
Jeffrey M. Nielsen
Marnie R. Salisbury
Randy Saunders
Roy Scrudder
David W. Seidel
Robert C. Turrell
Richard Weatherly
Annette Wilson

The working group acknowledges the following individuals who also contributed to the preparation of this standard:

Phillippe Annic
Tobin Bergen-Hill
Dominque Canazzi
Judith S. Dahmann
Peter Hoare
Mikael Karlsson
Michael Mazurek
J. Russell Noseworthy
Graham Shanks

The following members of the balloting committee voted on this standard:

Dale E. Anglin
Stephen Bachinsky
Michael R. Bachmann
David L. Barton
Brian Beebe
Robert P. Bennett
Christina L. Bouwens
Keith Briggs
Richard Briggs
Danny Cohen
Glenn E. Conrad
Mark Crooks
Dannie Cutts
Judith S. Dahmann
Steven Drake
Laura E. Feinerman
Jeff E. Fischer
Richard M. Fujimoto
Deborah Fullford
Brian F. Goldiez
Victor M. Gonzalez
James H. Hammond
Jack G. Harrington
Robert V. Head Jr.
Callie M. Hill
Ronald C. Hofer
James W. Hollenbach
Margaret M. Horst
Robert J. Howard
Ken Hunt
Kyle Isakson
James Ivers
Andreas Kemkes
John F. Kramer
Frederick S. Kuhl
Tom Lake
Gary M. Lightner
Paul R. Little
Reed Little
Bjorn Lofstrand
Margaret Loper
Robert R. Lutz
Jayne Lyons
Theodore F. Marz
Mark McAuliffe
David R. McKeeby
Duncan C. Miller
Katherine L. Morse
Thomas H. Mullins
Jeffrey M. Nielsen
Michael J. O'Conner
John Peng
Paul Perkinson
Hung Q. Phan
David R. Pratt
Paul M. Reeves
David Roberts
Marnie R. Salisbury
Randy Saunders
Roy Scrudder
David W. Seidel
Richard J. Severinghaus
Joseph F. Seward
Sean Sharp
Steven M. Sheasby
John W. Sheppard
Allen H. Skees
Col. Mark Smith
Susan D. Solick
Sandra Swearingen
Susan F. Symington
Donald Theune
William V. Tucker
Grant R. Tudor
John A. Tufarolo
Robert C. Turrell
Andrew J. Ventre
William F. Waite
Charles E. Walters
Richard Weatherly
Chris C. Wertman
Annette Wilson
Douglas D. Wood
Philomena M. Zimmerman
William H. Zimmerman

When the IEEE-SA Standards Board approved this standard on 21 September 2000, it had the following membership:

Donald N. Heirman, *Chair*
James T. Carlo, *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Mark D. Bowman
Gary R. Engmann
Harold E. Epstein
H. Landis Floyd
Jay Forster*
Howard M. Frazier
Ruben D. Garzon
James H. Gurney
Richard J. Holleman
Lowell G. Johnson
Robert J. Kennelly
Joseph L. Koepfinger*
Peter H. Lips
L. Bruce McClung
Daleep C. Mohla
James W. Moore
Robert F. Munzner
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Donald W. Zipse

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Alan Cookson, *NIST Representative*
Donald R. Volzka, *TAB Representative*

Jennifer McClain Longman
IEEE Standards Project Editor

Contents

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification

1. Overview

1.1 Scope

This document defines the format and syntax for recording information in High Level Architecture (HLA) object models, to include objects, attributes, interactions, and parameters. It does not define the specific data (e.g., vehicles, unit types) that will appear in the object models.

1.2 Purpose

The HLA has been developed to provide a common architecture for modeling and simulation. The HLA requires that federations (sets of federates) and individual federates (simulations, supporting utilities, or interfaces to live systems) be described by an object model that identifies the data exchanged at runtime to achieve federation objectives. This standard defines the documentation of the object model.

1.3 Background

1.3.1 Object model template rationale

A standardized structural framework, or template, for specifying HLA object models is an essential component of the HLA for the following reasons:

— It provides a commonly understood mechanism for specifying the exchange of data and general coordination among members of a federation.

— It provides a common, standardized mechanism for describing the capabilities of potential federation members.

— It facilitates the design and application of common tool sets for development of HLA object models.

HLA object models may be used to describe an individual federation member (federate), creating an HLA Simulation Object Model (SOM), or to describe a named set of multiple interacting federates (federation), creating a Federation Object Model (FOM). In either case, the primary objective of the HLA Object Model Template (OMT) is to facilitate interoperability among simulations and reuse of simulation components. All discussion of HLA object models in this document applies to both SOMs and FOMs unless explicitly stated otherwise.

1.3.2 Federation object models

During development of an HLA federation, it is critical that all federation members achieve a common understanding as to the nature or character of all required communications among participating federates. The primary purpose of an HLA FOM is to provide a specification for data exchange among federates in a common, standardized format. The content of this data includes an enumeration of all object and interaction classes pertinent to the federation and a specification of the attributes or parameters that characterize these classes. Taken together, the individual components of an HLA FOM establish the "information model contract" that is necessary (but not sufficient) to achieve interoperability among the federates.

1.3.3 Simulation object models

A critical step in the formation of a federation is the process of determining the composition of individual federates to best meet the sponsor's overall objectives. An HLA SOM is a specification of the types of information that an individual federate could provide to HLA federations and the information that an individual federate could receive from other federates in HLA federations. The standard format in which SOMs are expressed facilitates determination of the suitability of federates for participation in a federation.

The HLA OMT formats described in this document are generally applicable to either FOMs or SOMs. Thus, SOMs are also characterized by their objects, attributes, interactions, and parameters. The primary benefit from the common utilization of the OMT formats for FOMs and SOMs is that it provides a common frame of reference for describing object models in the HLA community. In some cases, this commonality may even allow SOM components to be integrated as "piece parts" in a FOM, facilitating FOM construction.

1.3.4 Relationship of HLA and object-oriented concepts

Although the HLA Object Model Template (OMT) is the standardized documentation structure for HLA object models, Federation Object Models (FOMs) and Simulation Object Models (SOMs) do not completely correspond to common definitions of object models in object-oriented (OO) analysis and design (OOAD) techniques. In the OOAD literature, an object model is described as an abstraction of a system developed for the purpose of fully understanding the system. To achieve this understanding, most OO techniques recommend defining several views of the system. For HLA object models, the intended scope of the system description is much narrower, focusing specifically on requirements and capabilities for federate information exchange. For SOMs, the intent is to describe the public interface of the federate in terms of an identified set of supported HLA object classes and interaction classes. A more complete description of how a federate is designed and functions internally (for example, a traditional OO object model) should be provided via documentation resources other than the SOM. For FOMs, the intent is to describe information exchange that happens during a federation execution.

Differences between HLA and OOAD principles and concepts also appear at the individual object level. In the OOAD literature, objects are defined as software encapsulations of data and operations (methods). In the HLA, objects are defined entirely by the identifying characteristics (attributes), values of which are exchanged between federates during a federation execution. Any OO-related behaviors and operations that affect the values of HLA object attributes are kept resident in the federates. Although HLA class structures are driven by subscription requirements and FOM growth concerns, class structures in OO systems are typically driven by efficiency and maintainability concerns.

As discussed above, HLA object classes are described by the attributes that are defined for them. These are, in OO parlance, data members of the class. These attributes, which are abstract properties of HLA object classes, are referred to as class attributes. HLA object instances are spawned via an HLA service using an

HLA object class as a template. Each attribute contained by an HLA object instance is called an instance attribute.

OO objects interact via message passing, in which one OO object invokes an operation provided by another OO object. HLA objects do not directly interact. It is the federates that interact, via HLA services, by updating instance attribute values or sending interactions. Also, responsibility for updating the instance attributes associated with an HLA object instance can be distributed among different federates in a federation (effectively distributing responsibility for maintaining the HLA object instance's state across the federation), whereas OO objects encapsulate state locally and associate update responsibilities with operations that are closely tied to the object's implementation in an OO programming language.

In addition to the stated semantic variations in shared terminology, other differences may also exist. Precise definitions of all HLA terms can be found in Clause 3.

2. References

This standard shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision shall apply:

ANSI Std X3.4-1986 (R1997), Information Systems—Coded Character Sets—7 Bit American National Standard Code Information Interchange (7-Bit ASCII).[1]

IEEE Std 754-1985 (R1990), IEEE Standard for Binary Floating-Point Arithmetic.[2]

IEEE Std 1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules.

IEEE Std 1516.1-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification.

REC-XML-19980210, Extensible Markup Language (XML) 1.0, W3C Recommendation, Feb. 1998.[3]

The Unicode Consortium, Ed. The Unicode Standard, Version 3.0. Reading, MA: Addison-Wesley Developers Press, Feb. 2000.

3. Definitions, abbreviations, acronyms, and conventions

3.1 Definitions

For the purposes of this standard, the following terms and definitions apply. The *Authoritative Dictionary of IEEE Standards Terms* [B2][4] should be referred for terms not defined in this clause.

3.1.1 accuracy: The measure of the maximum deviation of an attribute or a parameter value in the simulation or federation from reality or some other chosen standard or referent.

[1]ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

[3]This document is available at http://www.w3.org/.

[4]The numbers in brackets correspond to those of the bibliography in Annex F.

3.1.2 active subscription: A request to the runtime infrastructure (RTI) for the kinds of data (class attributes as well as interactions) that the joined federate is currently interested in receiving. The RTI uses this data along with publication data received from other joined federates to support services, such as

a) *Start/Stop Registration*
b) *Turn On/Off Updates*
c) *Turn On/Off Interactions*

The RTI also uses the subscription (and publication) data to determine how to route data to the joined federates that require that data.

3.1.3 attribute: A named characteristic of an object class or object instance. *See also*: **class attribute** and **instance attribute**.

3.1.4 attribute ownership: The property of an instance attribute that gives a joined federate the capability to supply values for that instance attribute to its federation execution. *See also*: **instance attribute**.

3.1.5 available attributes: The set of declared attributes of an object class in union with the set of inherited attributes of that object class.

3.1.6 available dimensions:

a) Pertaining to an attribute: the dimensions associated with the class attribute in the Federation Object Model (FOM). The available dimensions of an instance attribute are the available dimensions of the corresponding class attribute.
b) Pertaining to an interaction class: the dimensions associated with that interaction class in the FOM. The available dimensions of a sent interaction are the available dimensions of the interaction class specified in the *Send Interaction With Regions* service invocation.

NOTE—See 9.1 in IEEE Std 1516.1-2000.

3.1.7 available parameters: The set of declared parameters of an interaction class in union with the set of inherited parameters of that interaction class.

3.1.8 candidate discovery class: The registered class of an object instance, if subscribed. If the registered class of an object instance is not subscribed, the closest superclass of the registered class of the object instance to which the joined federate is subscribed. The term candidate discovery class pertains to object instances only.

3.1.9 candidate received class: The sent class of an interaction, if subscribed. If the sent class of an interaction is not subscribed, the closest superclass of the sent class of the interaction to which the joined federate is subscribed. The term candidate received class pertains to interactions only.

3.1.10 class: A description of a group of items with similar properties, common behavior, common relationships, and common semantics.

3.1.11 class attribute: A named characteristic of an object class denoted by a pair composed of an object class designator and an attribute designator.

3.1.12 class hierarchy: A specification of a class–subclass or "is–a" relationship between classes in a given domain.

3.1.13 coadunate: To attach an object instance handle (and possibly an object instance name) to an object instance.

3.1.14 collection: A set in which an element may occur multiple times (this corresponds to the mathematical notion of a bag).

3.1.15 collection of pairs: A group of pairs in which multiple pairs may have the same first component and a given pair may occur multiple times.

3.1.16 compliant object model: A High Level Architecture (HLA) Federation Object Model (FOM) or Simulation Object Model (SOM) that fully conforms with all of the rules and constraints specified in Object Model Template (OMT).

NOTE—See IEEE Std 1516.2-2000.

3.1.17 constrained set of pairs: A group of pairs in which no two pairs have the same first component (this corresponds to the mathematical notion of a function). An example would be a group of instance attribute and value pairs; each instance attribute may have at most one value associated with it.

3.1.18 corresponding class attribute of an instance attribute: The class attribute that, from the perspective of a given joined federate, is the class attribute of the joined federate's known class for the object instance containing the instance attribute that has the same attribute designator as the instance attribute.

3.1.19 corresponding instance attributes of a class attribute: The instance attributes that, from the perspective of a given joined federate, are

a) Unowned instance attributes of object instances that have a known class at the joined federate equal to the object class of the class attribute and that have the same attribute designator as the class attribute, or

b) Instance attributes owned by the joined federate that belong to object instances that have a known class at the owning federate equal to the object class of the class attribute and that have the same attribute designator as the class attribute.

3.1.20 datatype: A representation convention for a data element establishing its format, resolution, cardinality, and ordinality.

3.1.21 declared attributes: The set of class attributes of a particular object class that are listed in the Federation Object Model (FOM) as being associated with that object class in the object class hierarchy tree.

3.1.22 declared parameters: The set of parameters of a particular interaction class that are listed in the Federation Object Model (FOM) as being associated with that interaction class in the interaction class hierarchy tree.

3.1.23 default range: A range lower bound and a range upper bound, defined in the Federation Object Model Document Data (FDD) and specified in terms of [0, the dimension's upper bound), for a dimension.

3.1.24 default region: A multidimensional region provided by the runtime infrastructure (RTI) that is composed of one range for each dimension found in the Federation Object Model Document Data (FDD). The bounds of each of these ranges are [0, the range's dimension's upper bound). There is no way for a federate to refer to the default region.

NOTE—See 9.1 in IEEE Std 1516.1-2000.

3.1.25 designator: Some arguments (mostly identifiers) to services may have different views (implementations), depending on a particular programming language, application programmer's interface (API). For clarity, service descriptions refer to a generic view of the arguments, known as a *designator*.

3.1.26 dimension: A named interval. The interval is defined by an ordered pair of values, the first being the dimension lower bound and the second being the dimension upper bound. A runtime infrastructure (RTI) provides a predefined interval, whose lower and upper bounds are fixed as [0, the dimension's upperbound as specified in the Federation Object Model Document Data (FDD)]. This interval provides a single basis for communication of all dimension-related data with an RTI. All normalized intervals communicated to the RTI will be subsets of this interval.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.27 discover: To receive an invocation of the *Discover Object Instance †*[5] service for a particular object instance.

NOTE—See 6.5 of IEEE Std 1516.1-2000.

3.1.28 discovered class: The class that was an object instance's candidate discovery class at a joined federate when that object instance was discovered by that joined federate. *See also*: **candidate discovery class.**

NOTE—See 5.1.2 in IEEE Std 1516.1-2000.

3.1.29 exception: Notification of any irregularity that may occur during a service invocation.

3.1.30 federate: An application that may be or is currently coupled with other software applications under a Federation Object Model Document Data (FDD) and a runtime infrastructure (RTI). *See also*: **federate application** and **joined federate.**

3.1.31 federate application: An application that supports the High Level Architecture (HLA) interface to a runtime infrastructure (RTI) and that is capable of joining a federation execution. A federate application may join the same federation execution multiple times or may join multiple federation executions. However, each time a federate application joins a federation execution, it is creating a new joined federate. *See also*: **joined federate**.

3.1.32 federation: A named set of federate applications and a common Federation Object Model that are used as a whole to achieve some specific objective.

3.1.33 federation execution: The actual operation, over time, of a set of joined federates that are interconnected by a runtime infrastructure (RTI).

3.1.34 Federation Object Model (FOM): A specification defining the information exchanged at runtime to achieve a given set of federation objectives. This includes object classes, object class attributes, interaction classes, interaction parameters, and other relevant information.

3.1.35 Federation Object Model (FOM) Document Data (FDD): The data and information in an FOM document that is used by the *Create Federation Execution* service to initialize a newly created federation execution

NOTE—See IEEE Std 1516.2-2000.

3.1.36 handle: An identifier originated/created by the runtime infrastructure (RTI) that is federation execution-wide unique and unpredictable.

[5]All RTI initiated services are denoted with a † (printer's dagger) after the service name.

3.1.37 High Level Architecture (HLA) time axis: A totally ordered sequence of values in which each value typically represents an HLA instant of time in the physical system being modeled. For any two points T1 and T2 on the time axis, if T1 < T2, T1 represents an instant of time that occurs before the instant represented by T2.

3.1.38 in scope: Of or pertaining to an instance attribute of an object instance for which

a) The object instance is known to the joined federate

b) The instance attribute is owned by another joined federate, and

c) Either the instance attribute's corresponding class attribute is a

   1) Subscribed attribute of the known class of the object instance, or

   2) Subscribed attribute of the known class of the object instance with regions, and the update region set of the instance attribute at the owning joined federate overlaps the subscription region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate

   NOTE—See 6.1 of IEEE Std 1516.1-2000.

3.1.39 inherited attribute: A class attribute of an object class that was declared in a superclass of that object class in the object class hierarchy tree defined in the Federation Object Model (FOM).

3.1.40 inherited parameter: An interaction parameter that was declared in a superclass of that interaction class in the interaction class hierarchy tree defined in the Federation Object Model (FOM).

3.1.41 instance attribute: A named characteristic of an object instance denoted by a pair composed of the object instance designator and the attribute designator.

3.1.42 interaction: An explicit action taken by a federate that may have some effect or impact on another federate within a federation execution.

3.1.43 interaction class: A template for a set of characteristics that is common to a group of interactions. These characteristics correspond to the parameters that individual federates may associate with interactions.

3.1.44 interaction parameters: The information associated with an interaction that a federate potentially affected by the interaction may receive to calculate the effects of that interaction on its current state.

3.1.45 joined federate: A member of a federation execution, actualized by a federate application invoking the *Join Federation Execution* service as prescribed in IEEE Std 1516.1-2000. *See also*: **federate application**.

3.1.46 known class:

a) An object instance's registered class if the joined federate knows about the object instance as a result of having registered it, or

b) An object instance's discovered class if the joined federate knows about the object instance as a result of having discovered it.

3.1.47 known object instance: An object instance that a given joined federate has either registered or discovered, and one that the joined federate has not subsequently deleted (globally or locally) or was notified to remove. *See also*: **register** and **discover**.

3.1.48 logical time: A federate's current point on the High Level Architecture (HLA) time axis. Federates making use of the time management services follow restrictions on what time stamps can be sent in time stamp order (TSO) messages (relative to their logical time) to ensure that federates receiving those messages receive them in TSO.

3.1.49 lookahead: Lookahead is a nonnegative value that establishes a lower value on the time stamps that can be sent in time stamp order (TSO) messages by time-regulating joined federates. Once established, a joined federate's lookahead value may only be changed using the *Modify Lookahead* service. Each time-regulating joined federate must provide a lookahead value when becoming time-regulating.

3.1.50 Management Object Model (MOM): A group of predefined High Level Architecture (HLA) constructs (object and interaction classes) that provide the following:

a) Access to federation execution operating information
b) Insight into the operations of joined federates and the runtime infrastructure (RTI), and
c) Control of the functioning of the RTI, the federation execution, and the individual joined federates

3.1.51 message: A change of object instance attribute value, an interaction, or a deletion of an existing object instance, often associated with a particular point on the High Level Architecture (HLA) time axis, as denoted by the associated time stamp.

3.1.52 object class: A fundamental element of a conceptual representation for a federate that reflects the real world at levels of abstraction and resolution appropriate for federate interoperability. A template for a set of characteristics that is common to a group of object instances. These characteristics correspond to the class attributes that individual federates may publish and to which other federates may subscribe.

3.1.53 object instance: A unique instantiation of an object class that is independent of all other instances of that object class. At any point during a federation execution, the state of a High Level Architecture (HLA) object instance is defined as the collection of the values of all its instance attributes.

3.1.54 object model: A system specification defined primarily by class characteristics and relationships. The High Level Architecture (HLA) idea of an object model is similar in many ways, but not identical, to the common idea of an object model in object-oriented literature.

3.1.55 object model framework: The rules and terminology used to describe High Level Architecture (HLA) object models.

3.1.56 order type: A runtime infrastructure (RTI) provided means of ordering messages originating from multiple joined federates that are delivered to a single joined federate. Different categories of service are defined with different characteristics regarding whether and how an RTI orders messages that are to be delivered to a joined federate.

3.1.57 out of scope: Of or pertaining to an instance attribute of an object instance for which one or more of the following is not true:

a) The object instance is known to the joined federate
b) The instance attribute is owned by another joined federate, and
c) Either the instance attribute's corresponding class attribute is a
   1) Subscribed attribute of the known class of the object instance, or
   2) Subscribed attribute of the known class of the object instance with regions, and the update region set of the instance attribute at the owning joined federate overlaps the subscription

region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate.

NOTE—See 6.1 of IEEE Std 1516.1-2000.

3.1.58 overlap:

a) Pertaining to region sets: Two region sets overlap if there is a region in each set, such that the two regions overlap.

b) Pertaining to regions: If two regions have at least one dimension in common, they overlap if all ranges of dimensions that are contained in both regions overlap pairwise. If two regions do not have any dimensions in common, they do not overlap.

c) Pertaining to ranges: Two ranges A = $[a_{\text{lower}}, a_{\text{upper}})$ and B = $[b_{\text{lower}}, b_{\text{upper}})$ overlap, if and only if either $a_{\text{lower}} = b_{\text{lower}}$ or $(a_{\text{lower}} < b_{\text{lower}} < a_{\text{upper}})$.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.59 owned: Pertaining to the relationship between an instance attribute and the joined federate that has the unique right to update that instance attribute's value.

3.1.60 owned instance attribute: An instance attribute that is explicitly modeled by the owning joined federate. A joined federate that owns an instance attribute has the unique responsibility to provide values for that instance attribute to the federation, through the runtime infrastructure (RTI), as documented in the Federation Object Model Document Data (FDD).

3.1.61 pair: A grouping of two related elements (a first component and a second component), the combination of which is treated as an entity. An example of a pair would be an instance attribute grouped with its current value.

3.1.62 parameter: A named characteristic of an interaction.

3.1.63 passel: A group of attribute handle/value pairs from an *Update Attribute Values* service invocation that are delivered together via a *Reflect Attribute Values †* service invocation. All pairs within the passel have the same user-supplied tag, sent message order type, transportation type, receive message order type, time stamp (if present), and set of sent region designators (if present). A passel is a message.

3.1.64 passive subscription: A request to the runtime infrastructure (RTI) for the kinds of data (object classes and attributes as well as interactions) that the joined federate is currently interested in receiving, but unlike an active subscription, this information is not used by the RTI to arrange for data to be delivered, nor is it used to tell publishing joined federates that another joined federate is subscribing to that data (by way of *Start/Stop Registration*, *Turn On/Off Updates,* or *Turn On/Off Interactions* service invocations). This form of subscription is provided to support certain types of logger joined federates.

3.1.65 promoted: Pertaining to an object instance, as known by a particular joined federate, that has a discovered class that is a superclass of its registered class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.66 published:

a) Pertaining to an object class such that, from the perspective of a given joined federate, there is at least one available attribute of the object class that was an argument to a *Publish Object Class Attributes* service invocation that was not subsequently unpublished via the *Unpublish Object Class Attribut*es service.

NOTE—See 5.1.2 of IEEE Std 1516.1-2000.

b) Pertaining to an interaction class that, from the perspective of a given joined federate, was an argument to a *Publish Interaction Class* service invocation that was not subsequently followed by an *Unpublish Interaction Class* service invocation for that interaction class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.67 published attributes of an object class: The class attributes that have been arguments to *Publish Object Class Attributes* service invocations by a given joined federate for that object class that have not subsequently been unpublished (either individually or by unpublishing the whole object class), and possibly the **HLAprivilegeToDeleteObject** attribute for that object class.

NOTE—See 5.1.2 of IEEE Std 1516.1-2000.

3.1.68 range: A subset of a dimension, defined by an ordered pair of values, the first being the range lower bound and the second being the range upper bound. This pair of values defines a semi-open interval [range lower bound, range upper bound) (i.e., the range lower bound is the smallest member of the interval, and the range upper bound is just greater than any member of the interval).

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

3.1.69 range lower bound: The first component of the ordered pair of values that is part of a range.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.70 range upper bound: The second component of the ordered pair of values that is part of a range.

NOTE—See IEEE 9.1 of Std 1516.1-2000.

3.1.71 receive order (RO): A characteristic of no ordering guarantee for messages. Messages that are received as RO messages will be received in an arbitrary order by the respective joined federate. A time stamp value will be provided with the message if one was specified when the message was sent, but that time stamp has no bearing on message receipt order.

3.1.72 received class: The class that was an interaction's candidate received class at the joined federate when that interaction was received at that joined federate via an invocation of the *Receive Interaction* † service.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.73 received parameters: The set of parameters received when the *Receive Interaction †* service is invoked. These parameters consist of the subset of the sent parameters of an interaction that are available parameters of the interaction's received class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.74 reflect: Receive new values for one or more instance attributes via invocation of the *Reflect Attribute Values †* service.

NOTE—See 6.7 of IEEE Std 1516.1-2000.

3.1.75 reflected instance attribute: An instance attribute that is represented but not explicitly modeled in a joined federate. The reflecting joined federate accepts new values of the reflected instance attribute as they are produced by some other federation member and provided to it by the runtime infrastructure (RTI), via the *Reflect Attribute Values †* service.

3.1.76 region: A generic term that refers to either a region specification or a region realization. If not using Data Distribution Management (DDM), region arguments may be ignored.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.77 region realization: A region specification (set of ranges) that is associated with an instance attribute for update, with a sent interaction, or with a class attribute or interaction class for subscription. Region realizations are created from region specifications via the *Commit Region Modifications* (only when modifying a region specification from which region realizations are already derived), *Register Object Instance With Regions*, *Associate Regions for Updates*, *Subscribe Object Class Attributes With Regions*, *Subscribe Interaction Class With Regions*, *Send Interaction With Regions*, or *Request Attribute Value Update With Regions* services.

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

3.1.78 region specification: A set of ranges. Region specifications are created using the *Create Region* service, and a runtime infrastructure (RTI) is notified of changes to a region specification using the *Commit Region Modifications* service.

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

3.1.79 register: To invoke the *Register Object Instance* or the *Register Object Instance With Regions* service to create a unique object instance designator.

NOTE—See 6.4 of IEEE Std 1516.1-2000.

3.1.80 registered class: The object class that was an argument to the *Register Object Instance* or the *Register Object Instance With Regions* service invocation that resulted in the creation of the object instance designator for a given object instance.

3.1.81 resolution: The smallest resolvable value separating attribute or parameter values that can be discriminated. Resolution may vary with magnitude for certain data types.

3.1.82 retraction: An action performed by a federate to unschedule a previously scheduled message. Message retraction may be visible to the federate to whom the scheduled message was to be delivered. Retraction is widely used in classic event-oriented discrete event simulations to model behaviors such as preemption and interrupts.

3.1.83 runtime infrastructure (RTI) initialization data (RID): RTI vendor-specific information needed to run an RTI. If required, an RID is supplied when an RTI is initialized.

3.1.84 runtime infrastructure (RTI): The software that provides common interface services during a High Level Architecture (HLA) federation execution for synchronization and data exchange.

3.1.85 sent class: The interaction class that was an argument to the *Send Interaction* or *Send Interaction With Regions* service invocation that initiated the sending of a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.86 sent interaction: A specific interaction that is transmitted by a joined federate via the *Send Interaction* or *Send Interaction With Regions* service and received by other joined federates in the federation execution via the *Receive Interaction †* service.

3.1.87 sent parameters: The parameters that were arguments to the *Send Interaction* or *Send Interaction With Regions* service invocation for a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.88 set: A group of elements in which each element occurs at most once (this corresponds to the mathematical notion of sets). An example of a set would be a group of class attributes, each of which belongs to the same object class.

3.1.89 Simulation Object Model (SOM): A specification of the types of information that an individual federate could provide to High Level Architecture (HLA) federations as well as the information that an individual federate can receive from other federates in HLA federations. The standard format in which SOMs are expressed facilitates determination of the suitability of federates for participation in a federation.

3.1.90 specified dimensions: The dimensions that are explicitly provided when the region specification is created or modified.

NOTE—See 9.1.2 of IEEE Std 1516.1-2000.

3.1.91 stop publishing: To take action that results in a class attribute that had been a published attribute of a class no longer being a published attribute of that class.

3.1.92 subclass: A class that adds additional detail to (specializes) another more generic class (superclass). A subclass, by inheriting the properties from its parent class (closest superclass), also inherits the properties of all superclasses of its parent as well.

3.1.93 subscribed:

a) Pertaining to an object class for which, from the perspective of a given joined federate, there are subscribed attributes of that class or subscribed attributes of that class with regions, for some region. *See also*: **subscribed attributes of an object class** and **subscribed attributes of an object class with regions**.

b) Pertaining to an interaction class that is a subscribed interaction class or a subscribed interaction class with regions, for some region. *See also*: **subscribed interaction class** and **subscribed interaction class with regions**.

3.1.94 subscribed attributes of an object class: The class attributes that have been arguments to *Subscribe Object Class Attributes* service invocations by a given joined federate for a given object class that have not subsequently been unsubscribed, either individually or by unsubscribing the whole object class.

NOTE—See 5.1.2 and 5.6 of IEEE Std 1516.1-2000.

3.1.95 subscribed attributes of an object class with regions: The class attributes that have been arguments to *Subscribe Object Class Attributes With Regions* service invocations by a given joined federate for a given object class and a given region, assuming the joined federate did not subsequently invoke the *Unsubscribe Object Class Attributes With Regions* service for that object class and region.

NOTE—See 9.8 of IEEE Std 1516.1-2000.

3.1.96 subscribed interaction class: Pertaining to an interaction class that, from the perspective of a given joined federate, was an argument to a *Subscribe Interaction Class* or *Subscribe Interaction Class With Regions* service invocation that was not subsequently followed by an *Unsubscribe Interaction Class* or *Unsubscribe Interaction Class With Regions* service invocation for that interaction class.

NOTE—See 5.1.3 and 5.8 of IEEE Std 1516.1-2000.

3.1.97 subscribed interaction class with regions: Pertaining to an interaction class and a region that, from the perspective of a given joined federate, were arguments to a *Subscribe Interaction Class With Regions* service invocation that was not subsequently followed by an *Unsubscribe Interaction Class With Regions* service invocation for that interaction class and that region.

NOTE—See 9.10 of IEEE Std 1516.1-2000.

3.1.98 subscription region set: A set of regions used for subscription of a class attribute or used for subscription of an interaction class. *See also*: **used for subscription of a class attribute** and **used for subscription of an interaction class**.

3.1.99 superclass: A class that generalizes a set of properties that may be inherited by more refined (i.e., detailed) versions of the class. In High Level Architecture (HLA) applications, a class may have at most one immediate superclass (i.e., can only inherit from a single class at the next highest level of the class hierarchy).

3.1.100 synchronization point: A logical point in the sequence of a federation execution that all joined federates forming a synchronization set for that point attempt to reach and, if they are successful, thereby synchronize their respective processing at that point.

3.1.101 time advancing state: A joined federate may advance its logical time only by requesting a time advancement from the runtime infrastructure (RTI) via one of the following services:

a) *Time Advance Request*
b) *Time Advance Request Available*
c) *Next Message Request*
d) *Next Message Request Available*
e) *Flush Queue Request*

The joined federate's logical time will not actually be advanced until the RTI responds with a *Time Advance Grant †* service invocation at that joined federate. During the interval between a request to advance its logical time and the corresponding grant, the joined federate is in the time advancing state.

3.1.102 time-constrained federate: A joined federate that may receive time stamp order (TSO) messages and whose time advances are constrained by other joined federates within a federation execution.

NOTE—See 8.1 of IEEE Std 1516.1-2000.

3.1.103 time-regulating federate: A joined federate that may send time stamp order (TSO) messages and that constrains the time advances of other joined federates within a federation execution.

NOTE—See 8.1 of IEEE Std 1516.1-2000.

3.1.104 time management: A collection of High Level Architecture (HLA) services that support controlled message ordering and delivery to the cooperating joined federates within a federation execution in a way that is consistent with federation requirements.

3.1.105 time stamp (of message or save): The value of the time stamp argument provided to the relevant service invocation.

3.1.106 time stamp order (TSO): An ordering of messages provided by a runtime infrastructure (RTI) for joined federates making use of time management services and messages containing time stamps. Messages having different time stamps are said to be delivered in TSO if for any two messages M1 and M2 (time stamped with T1 and T2, respectively) that are delivered to a single joined federate where T1 < T2, then M1 is delivered before M2. Messages having the same time stamp will be delivered in an arbitrary order (i.e., no tie-breaking mechanism is provided by an RTI).

3.1.107 transportation type: A runtime infrastructure (RTI) provided means of transmitting messages between joined federates. Different categories of service are defined with different characteristics such as reliability of delivery and message latency.

3.1.108 unspecified dimensions: The available dimensions of a class attribute, instance attribute, interaction class, or sent interaction less the specified dimensions of the region specification from which the region realization is derived.

NOTE—See 9.1.3 of IEEE Std 1516.1-2000.

3.1.109 update: Invoke the *Update Attribute Values* service for one or more instance attributes.

NOTE—See 6.6 of IEEE Std 1516.1-2000.

3.1.110 update region set: A set of regions used for sending interactions or used for update of instance attributes. *See also*: **used for sending** and **used for update**.

3.1.111 used for sending:

a) Pertaining to a region that, along with the specified interaction class designator, is being used as an argument in the *Send Interaction With Regions* service.

b) Pertaining to the default region when the specified interaction class designator is being used as an argument in the *Send Interaction* service.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.112 used for subscription of a class attribute:

a) Pertaining to a region, an object class, and a class attribute for which the class attribute is a subscribed attribute of the object class with that region. *See also*: **subscribed attributes of an object class with regions.**

   NOTE—See 9.1 of IEEE Std 1516.1-2000.

b) Pertaining to the default region when the specified class attribute is a subscribed attribute of the specified class. *See also*: **subscribed attributes of an object class with regions.**

   NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.113 used for subscription of an interaction class:

a) Pertaining to a region and an interaction class for which the interaction class is a subscribed interaction class with regions. *See also*: **subscribed interaction class**.

   NOTE—See 9.1 of IEEE Std 1516.1-2000.

b) Pertaining to the default region when the specified interaction class is a subscribed interaction class. *See also*: **subscribed interaction class**.

   NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.114 used for update:

a) Pertaining to a region that, along with the specified object instance and attribute designators, has been used as an argument in either the *Register Object Instance With Regions* service or the *Associate Regions For Updates* service; and the region has not subsequently been used along with the specified object instance designator as an argument in the *Unassociate Regions For Updates* service; nor has the joined federate subsequently lost ownership of the specified instance attribute(s).

NOTE—See 9.1 of IEEE Std 1516.1-2000.

b) Pertaining to the default region when the specified instance attribute(s) is not currently used for update with any other region.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.2 Abbreviations and acronyms

The following abbreviations and acronyms pertain to this standard.

API	application programmer's interface
BNF	Backus Naur Form
DDM	Data Distribution Management
DIF	data interchange format
DM	Declaration Management
DTD	Document Type Declaration
FDD	FOM Document Data
FOM	Federation Object Model
FQR	*Flush Queue Request*
GALT	Greatest Available Logical Time
HLA	High Level Architecture
LITS	Least Incoming Time Stamp
MOM	Management Object Model
M&S	modeling and simulation
NA	not applicable
NMR	*Next Message Request*
NMRA	*Next Message Request Available*
OMT	Object Model Template
OO	object oriented
OOAD	object-oriented analysis and design
POC	point of contact
RID	RTI initialization data
RO	receive order
RTI	runtime infrastructure
SISC	Simulation Interoperability Standards Committee
SOM	Simulation Object Model
TAR	*Time Advance Request*
TARA	*Time Advance Request Available*
TRADT	time representation abstract datatype
TSO	time stamp order
XML	eXtensible Markup Language

3.3 Conventions

Conventions in this standard specify how entries are to be formulated when completing an HLA object model. The following conventions pertain to this standard.

3.3.1 Names

Names in object models shall adhere to XML naming conventions. XML conventions require that names be constructed from a combination of letters, digits, hyphens, colons, full stops (periods), and underscores with no spaces or other breaking characters (e.g., tabs, carriage returns, etc.). In HLA object models, allowable names shall be further restricted as follows:

a) The period (full stop) is reserved for qualifying class names and shall not be included in a user-defined name within an object model. When fully qualified, the class name includes all predecessor class names (separated by periods) beginning at the root and ending at the class name in question.

b) As recommended in the XML specification, the colon character shall not be used.

c) Names beginning with the string “hla,” or any string that would match (('H'|'h') ('L'|'l') ('A'|'a')), are reserved and shall not be included in user-defined names.

d) The case of all textual data within HLA object models (including names) shall be significant; all textual data defined in an object model document shall be case-sensitive.

e) A name consisting of the string “na,” or any string that would match (('N'|'n') ('A'|'a')), is reserved to indicate that a name is not applicable in this circumstance and shall not be included as a user-defined name.

These rules apply to the following names in HLA object models:

— Object class names
— Interaction class names
— Attribute names
— Parameter names
— Datatype names
— Enumerated datatype enumerators
— Enumerated datatype values
— Fixed record field names
— Variant record alternative names
— Basic data representation names
— Dimension names
— Transportation type names
— Synchronization point labels
— Note identifying labels

4. HLA OMT components

HLA object models are composed of a group of interrelated components specifying information about classes of objects and their attributes and interactions and their parameters. The information content of these components can be represented in many different ways or presentations. A presentation is the formatting of the information contained in the object model in a particular manner for a particular purpose. For example, the OMT tabular format is designed for presentation on a printed page, whereas the OMT DIF is a presentation designed for passing an object model between tools. All HLA object models shall be capable of being presented in both the OMT tabular format and the OMT DIF format. This document defines the OMT content and presents it in OMT tabular format throughout.

The OMT consists of the following components:

— *Object model identification table*: To associate important identifying information with the HLA object model

— *Object class structure table*: To record the namespace of all federate or federation object classes and to describe their class-subclass relationships

— *Interaction class structure table*: To record the namespace of all federate or federation interaction classes and to describe their class–subclass relationships

— *Attribute table*: To specify features of object attributes in a federate or federation

— *Parameter table*: To specify features of interaction parameters in a federate or federation

— *Dimension table*: To specify dimensions for filtering instance attributes and interactions

— *Time representation table*: To specify the representation of time values

— *User-supplied tag table*: To specify the representation of tags used in HLA services

— *Synchronization table*: To specify representation and datatypes used in HLA synchronization services

— *Transportation type table*: To describe the transportation mechanisms used

— *Switches table*: To specify initial settings for parameters used by the RTI

— *Datatype tables*: To specify details of data representation in the object model

— *Notes table*: To expand explanations of any OMT table item

— *FOM/SOM lexicon*: To define all of the objects, attributes, interactions, and parameters used in the HLA object model

All of the OMT components shall be completed when specifying an HLA object model for both federations and individual federates. However, certain tables may be empty or devoid of domain-specific content. For instance, although federations typically support interactions among their federates, some federates (such as a stealth viewer) might not be involved in interactions. In this situation, the interaction class structure table would contain only the single interaction class required by the HLA and the parameter table would be empty in that federate's SOM. It is also expected that federates commonly have objects with attributes of interest across the federation; in such cases, these objects and attributes shall be documented. However, a federate or an entire federation may exchange information solely via interactions; in which case, its object class structure table and attribute table would contain only HLA-required data. The specific rules for the applicability of each OMT table are provided in the table descriptions.

The final HLA OMT component, the FOM/SOM lexicon, is essential to ensure that the semantics of the terms used in an HLA object model are understood and documented.

The HLA MOM specifies a designated set of information elements that are associated with federation executions. Implementation of the MOM information elements as specified in IEEE Std 1516.1-2000 provides a mechanism for management of federation executions using existing HLA services. Inclusion of the MOM is required for all FOMs. In addition, federates that have an inherent capability to interact with or extend the MOM shall include the MOM in their SOM. Inclusion of the MOM means the inclusion of all tables in Clause 11 of IEEE Std 1516.1-2000, along with any applicable MOM extensions.

Any FOM or SOM that fully conforms to all of the rules and constraints stated in this specification is a *compliant object model*.

The basics of each OMT component are presented in the following separate subclauses. The template format for each component is provided and described, and criteria are suggested to help guide decisions on when to include specific federate or federation features within each of these components for a specific HLA object model. Examples of the usage of each OMT component are also provided; these examples are for illustrative purposes only, and they are not meant to imply any additional requirements. Annex A specifies the notations used in describing OMT tables.

4.1 Object model identification table

4.1.1 Purpose/background

A key design goal for all HLA object models is to facilitate reuse. HLA object models provide information that enables inferences to be drawn regarding the reuse potential of individual federates for new applications. Reuse can also occur at the level of the object model. An existing FOM may provide a foundation for the development of new FOMs, or applicable components of existing SOMs may be merged to form new FOMs. In either case, to expedite reuse, it is important to include a minimum but sufficient degree of descriptive information in the object model. For instance, when federation developers wish to pose detailed questions about how a federate or federation was constructed, POC information within an HLA object model is extremely important. The purpose of this table is to document certain key identifying information within the object model description.

4.1.2 Table format

The information that identifies HLA object models shall be provided in a simple two-column table. The structure to be used to describe this information is provided in Table 1.

The first column (Category) specifies the categories of data that shall be provided in this table. The second column (Information) shall specify the required information. Entries shall be provided for all rows except References and Other; "NA" shall be entered for these rows if no information is appropriate. Definitions of the categories and required information are as follows:

a) *Name*: This field shall specify the name assigned to the object model.

b) *Type*: This field shall specify the type that the object model represents; valid values are

   — *FOM*: the object model describes a federation

   — *SOM*: the object model describes a federate

c) *Version*: This field shall specify the version identification assigned to the object model.

d) *Modification Date*: This field shall specify the latest date on which this version of the object model was created or modified. The modification date shall be specified in the format "YYYY-MM-DD" (e.g., 1999-04-15).

e) *Purpose*: This field shall specify the purpose for which the federate or federation was developed.

f) *Application Domain*: This field shall specify the type or class of application to which the federate or federation applies. The following values may be used for this field, although other values are also valid:

— Analysis

— Training

— Test and Evaluation

— Engineering

— Acquisition

g) *Sponsor*: This field shall specify the organization that sponsored the development of the federate or federation

h) *POC*: This field shall specify the point of contact for information on the federate or federation and the associated object model, including an honorific (e.g., Dr., Ms., etc.) or rank, first name, and last name where appropriate

i) *POC Organization*: This field shall specify the organization with which the POC is affiliated

j) *POC Telephone*: This field shall specify the telephone number for the POC, including the international telephone code for the POC's country

k) *POC E-mail*: This field shall specify the e-mail address of the POC

l) *References*: This field shall specify pointers to additional sources of information. For example, documents that describe save/restore semantics, time management strategies, or other relevant federation policy decisions may be identified through this field. "NA" shall be entered when no such references are appropriate.

m) *Other*: This field shall specify other data deemed relevant by the author of the object model. "NA" shall be entered when no such data are appropriate.

Table 1—Object model identification table

Category	Information
Name	<name>
Type	<type>
Version	<version>
Modification Date	<date>
Purpose	<purpose>
Application Domain	<application domain>
Sponsor	<sponsor>
POC	<poc>
POC Organization	<poc organization>
POC Telephone	<poc telephone>
POC Email	<poc email>
References	<references>
Other	<other>

4.1.3 Inclusion criteria

The categories of information specified in this table shall be included for all HLA FOMs and SOMs.

4.1.4 Example

Table 2 shows a simple example of the object model identification table. In this example, the *Purpose* entry is shorter (by design) than entries that would be found in this table for most practical applications. The subclauses that follow provide examples of each OMT table for the notional SOM identified in this table.

Table 2—Object model identification table example

Category	Information
Name	RestaurantExample
Type	SOM
Version	1.0 Alpha
Modification Date	1998-01-01
Purpose	Example of an object model for a restaurant federate
Application Domain	Restaurant operations
Sponsor	Federated foods
POC	Mr. Joseph Doe
POC Organization	Joe's Place
POC Telephone	1-977-555-1234
POC E-mail	doej@fedfoods.com
References	www.fedfoods.com/restsim.html
Other	See Mobil International Restaurant Guide for more information

4.2 Object class structure table

4.2.1 Purpose/background

The object class structure of an HLA object model is defined by a set of relations among classes of objects from the simulation or federation domain. An HLA object class is a collection of objects with certain characteristics or attributes in common. Each of the individual objects that are realizations of a class is said to be an instance of that class.

An HLA object class structure shall be defined by hierarchical relationships among classes of objects. Class relationships shall be represented by the inclusion of the associated class names in the appropriate columns of the object class structure table. Relationships among classes at nonadjacent levels of a class hierarchy can be derived through transitivity: If A is a superclass of B, and B is a superclass of C, A is also a superclass of C. Superclass and subclass play inverse roles in these relations: If A is a superclass of B, B is a subclass of A.

Subclasses can be considered to be specializations, or refinements, of their superclasses. Subclasses shall always inherit the attributes of their superclasses, and they may possess additional attributes to provide the desired specialization. For instance, suppose a "Circle" class is defined as a subclass of a "Figure" class. Attributes defined for the Figure class are those that apply to all types of figures, such as "Color" or "Line Thickness." The Circle subclass automatically inherits the attributes of the Figure class; however, an additional "Radius" attribute might also be defined that is only appropriate for this special type of figure.

A class is a leaf of a class structure if it has no subclasses. The object class “HLAobjectRoot” shall be a superclass of all other object classes in a FOM or SOM. The HLA object model shall support only single inheritance; in this mechanism, each class has at most one immediate superclass.

HLA defines three types of object class attributes:

— **declared attributes:** The set of class attributes of an object class that are associated with that object class in the object class hierarchy tree.
— **inherited attributes:** The set of class attributes of an object class that was declared in a superclass of that object class in the object class hierarchy tree.
— **available attributes:** The set of declared attributes of an object class in union with the set of inherited attributes of that object class.

Federates participating in a federation execution may subscribe to attributes of object classes at any level of the class hierarchy. Conditions and circumstances surrounding discovery and reflection of instance attributes are described in IEEE Std 1516.1-2000.

Object classes provide the means for federation participants to subscribe to information about all individual instances of HLA objects with common characteristics, such as all M1A1 tanks or F117A fighters. Classes are also essential to specifying characteristics (attributes) of simulation objects, because these are defined relative to classes of objects, not unique to individual instances. In addition, because basic HLA services (as described in IEEE Std 1516.1-2000) support subscriptions to object class attributes by federates participating in a federation execution, the RTI requires knowledge of all object classes and their attributes if it is to support distribution of HLA object information by class to the federates of a federation execution.

A class hierarchy expands the capabilities of a flat classification scheme by enabling federates to subscribe to information about broad superclasses of objects, such as all tanks, all attack fighters, or even all ground vehicles, air vehicles, or sea vehicles. IEEE Std 1516.1-2000 supports subscription to attributes of any class in an object class hierarchy so that federates can easily subscribe to attributes of all or only those classes of interest. An object class hierarchy also supports simplification of the specification of attributes by placing common attributes of multiple subclasses in a common superclass. Thus, class hierarchies enable simpler management of the interests of different federates in the objects and attributes involved in a federation execution.

4.2.2 Table format

The object class structure template of Table 3 provides a format for representing the class–subclass hierarchy of object classes. It shall be populated by first entering the root class of all object classes in the left-most column; this object class is named “HLAobjectRoot.” Then, the most general object classes shall be entered in the next column, followed by all of their immediate subclasses in the next column, and then further levels of subclasses, as required. The number of intermediate columns used depends on the needs of the federate or federation. A federate or federation that uses a deeper hierarchy than illustrated by the template of the table shall add columns as needed. Finally, the most specific object classes shall be specified in the right-most column. Object class names shall adhere to HLA naming conventions (see 3.3.1).

Although individual class names need not be unique, all object classes shall be uniquely identifiable in an HLA object model when concatenated (via dot notation) with the names of higher level superclasses.

Each object class in the object class structure table shall be followed by information on publication and subscription capabilities enclosed in parentheses, as designated in the template using the abbreviated variable name *<p/s>*.

For a SOM, valid entries for *<p/s>* shall be as follows:

— *P (Publish)*: The federate is capable of publishing at least one attribute of the object class.
— *S (Subscribe)*: The federate is capable of subscribing to at least one attribute of the object class.
— *PS (PublishSubscribe)*: The federate is capable of publishing at least one attribute and subscribing to at least one attribute of the object class.
— *N (Neither)*: The federate is incapable of either publishing or subscribing to any attributes of the object class.

For a FOM, the same entries for *<p/s>* are valid. However, an object class is classified as *Publish* or *Subscribe* in the federation if at least one federate is capable of publishing or subscribing to at least one attribute in the context of the object class.

Classes designated as *Subscribe* or *Neither* can have no registered instances, but they can have subclasses that can be registered.

Table 3—Object class structure table

<table>
<tr><td rowspan="12">HLAobject Root (<p/s>)</td><td rowspan="8">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td rowspan="3">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td rowspan="3">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td rowspan="3">[<class> (<p/s>)]</td><td rowspan="3">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td>⋮</td><td>⋮</td><td>⋮</td><td>…</td><td>⋮</td></tr>
</table>

4.2.3 Inclusion criteria

In all HLA object models, any object class that is referenced in any table within the object model shall also be included in the object class hierarchy. The criteria for designing an object class hierarchy for an HLA object model is fundamentally different for individual federates than for federations. The object class structure table of a FOM represents an agreement between the federates in a federation on how to classify object classes for the purposes of federation executions. The object class structure table of a SOM is a type of advertisement of the classes of objects that the federate can support (as *Publish* or *Subscribe*) in potential federations. In neither case does the HLA require specific object classes (other than the required root class) or object class hierarchies to appear in the object class structure table.

For individual federates, it is the intrinsic functionality (expressed as classes of objects) that the federate can offer to future HLA federations—and any object classes whose instances (and associated instance attribute values) may potentially represent useful information if generated by other HLA federates—that defines the content of the object class structure table. The structure of the object class hierarchy is driven by how the federate supports class publication and subscription. Rich, deep SOM class hierarchies can provide federates with a significant degree of flexibility in how they can support and participate in federations in the future.

For federations, it is the subscription requirements of the collective set of simulations participating in the federation that drives the content and structure of the object class hierarchy. Although a set of object classes for the most specific types of entities involved in a federation (e.g., M1 tanks and Bradley fighting vehicles) may completely satisfy the subscription requirements of some types of HLA applications, additional higher level object classes will be needed if federates wish to be able to subscribe to HLA object information at higher levels of abstraction (e.g., tanks, armored vehicles, or ground vehicles). For a federate to be able to subscribe to HLA object information at a desired level of abstraction, an object class at that level of abstraction shall appear in the object class structure table.

For example, suppose a federation involved air, land, and sea forces of many specific types. If a particular federate did not require notification of the specific types of land vehicles, but did require notification of land vehicles in its area of interest, a suitable class (such as *GroundVehicle*) that contained specific vehicle types as subclasses would be needed to make this possible.

Although classes are clearly needed for all objects of interest in a federation, many alternative class hierarchies can be devised to cover any given set of objects. The demarcations and levels of classes selected for an HLA FOM are the result of the federation development process. Object class hierarchies that already exist for individual SOMs may be incorporated into a FOM object class hierarchy if they meet the interests of the federation as a whole.

4.2.4 Example

Table 4 provides an example of how the object class structure table can be used to represent a simple system. In this case, the system being represented is a typical neighborhood restaurant. The simulation of this restaurant's operations can be considered to be a potential federate in a larger scale federation, perhaps representing the combined, coordinated operation of a chain of restaurants. The intent of this example is not to specify a complete SOM for this system, but rather to provide partial illustrations of how the OMT tables can be used to capture relevant information about the system.

In this example, a subset of a complete object class hierarchy is shown as consisting of five object classes at the most general level. For this simulation, no class decomposition was necessary for the first three classes. For the fourth class, a single level of decomposition is shown resulting in five leaf classes. For the fifth class, several levels of decomposition are shown to illustrate a partial representation of the restaurant's menu. Some of the deeper levels in this hierarchy could have been modeled as attributes (e.g., *ClamChowder* could have been a leaf node, with an attribute of *Type* to represent the enumerated values of *Manhattan* or *NewEngland*). However, the modeler in this example opted to represent the most specific food types as individual classes.

In this example, most nonleaf classes are designated as *Subscribe* only, whereas the leaf nodes (concrete classes) are designated as both *Publish* and *Subscribe*. One exception to this is the *Employee* class which contains several attributes that are only published or subscribed to at the subclass level. Another exception is the *ClamChowder* class, which provides an example of a nonleaf class that can be registered (i.e., has attributes that can be published for that class). Finally, *Manhattan* and *NewEngland* provide examples of publishable leaf classes that do not contain subscribable attributes.

Table 4—Object class structure table example

<table>
<tr><td rowspan="28">HLA object Root (N)</td><td>Customer (PS)</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Bill (PS)</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Order (PS)</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">Employee (N)</td><td>Greeter (PS)</td><td></td><td></td><td></td></tr>
<tr><td>Waiter (PS)</td><td></td><td></td><td></td></tr>
<tr><td>Cashier (PS)</td><td></td><td></td><td></td></tr>
<tr><td>Dishwasher (PS)</td><td></td><td></td><td></td></tr>
<tr><td>Cook (PS)</td><td></td><td></td><td></td></tr>
<tr><td rowspan="20">Food (S)</td><td>MainCourse (PS)</td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">Drink (S)</td><td>Water (PS)</td><td></td><td></td></tr>
<tr><td>Coffee (PS)</td><td></td><td></td></tr>
<tr><td>Soda (PS)</td><td></td><td></td></tr>
<tr><td rowspan="4">Appetizers (S)</td><td rowspan="3">Soup (S)</td><td rowspan="2">ClamChowder (PS)</td><td>Manhattan (P)</td></tr>
<tr><td>NewEngland (P)</td></tr>
<tr><td>BeefBarley (PS)</td><td></td></tr>
<tr><td>Nachos (PS)</td><td></td><td></td></tr>
<tr><td rowspan="6">Entree (S)</td><td>Beef (PS)</td><td></td><td></td></tr>
<tr><td>Chicken (PS)</td><td></td><td></td></tr>
<tr><td rowspan="3">Seafood (S)</td><td>Fish (PS)</td><td></td></tr>
<tr><td>Shrimp (PS)</td><td></td></tr>
<tr><td>Lobster (PS)</td><td></td></tr>
<tr><td>Pasta (PS)</td><td></td><td></td></tr>
<tr><td rowspan="3">SideDish (S)</td><td>Corn (PS)</td><td></td><td></td></tr>
<tr><td>Broccoli (PS)</td><td></td><td></td></tr>
<tr><td>BakedPotato (PS)</td><td></td><td></td></tr>
<tr><td rowspan="3">Dessert (S)</td><td>Cake (PS)</td><td></td><td></td></tr>
<tr><td rowspan="2">IceCream (S)</td><td>Chocolate (PS)</td><td></td></tr>
<tr><td>Vanilla (PS)</td><td></td></tr>
</table>

4.3 Interaction class structure table

4.3.1 Purpose/background

An interaction is defined as an explicit action taken by a federate that may have some effect or impact on another federate within a federation execution. Interactions shall be specified in the interaction class structure table of HLA object models in terms of their class–subclass relationships, in much the same way that objects are described in the object class structure table. The hierarchical structure of interactions supported by this table is composed of relations of generalization (or specialization) between different types of interactions. For example, an engagement interaction might be specialized by air-to-ground engagements, ship-to-air engagements, and others. This engagement interaction, then, would be said to generalize its more specific types.

An interaction hierarchy in an HLA FOM is primarily designed to support inheritance. In SOMs, an interaction hierarchy reflects how the federate supports publication and subscription of interaction classes.

The interaction class "HLAinteractionRoot" shall be a superclass of all other interaction classes in a FOM or SOM. The HLA object model shall support only single inheritance; in this mechanism, each class has at most one immediate superclass. Conditions and circumstances surrounding the processing of interactions are described in IEEE Std 1516.1-2000.

Interactions are one of the principal determinants of interoperability among simulations. Interoperability ordinarily requires some consistency in the treatment of interactions afforded by the different federates in which they appear. In distributed war fighting, for example, some uniformity in treatment of engagement interactions is commonly required to ensure a fair fight between objects simulated by different federates. Thus, all interactions in a FOM shall be identified, and all federates in an HLA federation must treat the specified interactions in a consistent fashion.

In addition, the types of interactions involved in a simulation execution shall be made known to the RTI in order to support publication and subscription to their occurrences. Thus, the HLA object model shall document all of the interactions that may be sent during a federation execution so that the RTI can recognize them.

4.3.2 Table format

The template that shall be used for recording HLA interactions for a federation or an individual federate is illustrated in Table 5. The basic format for this table follows the format specified earlier for the object class structure table. Thus, the root interaction class shall be specified in the left-most column and shall be named "HLAinteractionRoot." Subsequent columns to the right shall contain interaction classes with increasing degrees of class specificity. Like the object class structure table, additional columns may be added to the table as needed to specify the full hierarchy. Although individual class names need not be unique, all interaction classes shall be uniquely identifiable in an HLA object model when concatenated (via dot notation) with the names of higher level superclasses. Interaction class names shall adhere to HLA naming conventions (see 3.3.1).

Table 5—Interaction class structure table

<table>
<tr><td rowspan="12">HLAinteraction Root (<p/s>)</td><td rowspan="8">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td rowspan="3">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td rowspan="3">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td rowspan="3">[<class> (<p/s>)]</td><td rowspan="3">[<class> (<p/s>)]</td><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>[<class> (<p/s>)]</td><td>…</td><td>[<class> (<p/s>)]</td></tr>
<tr><td>⋮</td><td>…</td><td>⋮</td></tr>
<tr><td>⋮</td><td>⋮</td><td>⋮</td><td>…</td><td>⋮</td></tr>
</table>

Each interaction class in the interaction class structure table shall be followed by information on publishing and subscribing capabilities enclosed in parentheses, as designated in the template using the abbreviated variable name <*p/s*>.

For a SOM, valid entries for <*p/s*> shall be as follows:

— *P (Publish)*: The federate is capable of publishing the interaction class.

— *S (Subscribe)*: The federate is capable of subscribing to the interaction class.

— *PS (PublishSubscribe)*: The federate is capable of publishing and subscribing to the interaction class.

— *N (Neither)*: The federate is incapable of either publishing or subscribing to the interaction class.

For a FOM, the same entries for <*p/s*> are valid. An interaction class is classified as *Publish* or *Subscribe* in the federation if at least one federate is capable of publishing or subscribing to the class.

Classes designated as *Subscribe* or *Neither* are never sent, but they can have subclasses that are sent.

4.3.3 Inclusion criteria

A type of interaction shall be included in a FOM whenever it can take place "across" a federation, i.e., when it is published by one federate and subscribed to by another. Common examples of such interactions in warfighting simulations include a variety of engagement interactions between platforms that may be simulated by different federates. In order to document the types of interactions that federation members may need to accommodate, a FOM shall include all cross-federate interactions.

When interactions are expected to occur within an individual federate, they need not appear in a FOM. For example, in an engineering simulation, the interactions involved in the internal dynamics of a vehicle engine should not be part of a FOM if only one federate in the federation will interact directly with the engine component.

Because HLA SOMs are intended to be developed independently of any particular federation application, the relevance of any currently supported interaction class to future federations will generally be unknown. Thus, a simulation that supports either publishing or subscribing for an interaction class should document that support in its SOM if it might be of interest to future federations.

4.3.4 Example

A representation of some illustrative interactions, based on the restaurant example introduced in 4.2.4, is given in Table 6. Here, the operations of the restaurant are described according to a set of interactions between customers and the employees of the restaurant. At the highest level, the restaurant federate can publish generic customer–employee transactions. This would, for instance, allow federates that simulate management operations across the restaurant chain to subscribe to and monitor general activity levels for individual restaurants. This single high-level interaction class is then decomposed into the basic types of customer–employee interactions that occur in the restaurant. Although all classes at this level can be published by the federate, *OrderTaken*, *FoodServed*, and *CustomerPays* are all further decomposed into more specialized classes that can also be published by the restaurant federate depending on the needs of the federation. In addition, this federate also shows the ability to subscribe to (*S* designation) interactions of classes *CustomerSeated* and *CustomerLeaves*, possibly to monitor customer arrival activity in other restaurants within the chain, and to monitor the rate at which other restaurants can service their customers.

Table 6—Interaction class structure table example

HLAinteractionRoot (N)	CustomerTransaction (P)	CustomerSeated (PS)	
		OrderTaken (P)	FromKidsMenu (P)
			FromAdultMenu (P)
		FoodServed (P)	DrinkServed (P)
			AppetizerServed (P)
			MainCourseServed (P)
			DessertServed (P)
		CustomerPays (P)	ByCreditCard (P)
			ByCash (P)
		CustomerLeaves (PS)	

4.4 Attribute table

4.4.1 Purpose/background

Each class of simulation domain objects shall be characterized by a fixed set of attribute types. These attributes are named portions of their object's state whose values can change over time (such as location or velocity of a platform). The values of HLA instance attributes are updated through the RTI and provided to other federates in a federation. Both federates and federations shall document all such object attributes in the attribute table of their SOM or FOM. The object class "HLAobjectRoot" shall be a superclass of all other object classes in a FOM or SOM; attributes may be assigned to it like any other object class.

Attributes of HLA object classes are specified in order to support subscription to their values by other interested members of a federation. Thus, the names of attributes and associated object classes are essential information when initializing a federation execution. Knowledge of object attributes is commonly required for effective communication between federates in a federation. In addition, although the datatypes and update policies of attributes represent characteristics that are not directly utilized by the RTI, they are important to ensuring compatibility among federates in a federation. A federate operating with very low update rates for an instance attribute could create problems for interacting federates that are operating at high update rates. The specification of datatypes and update rates in an HLA FOM is a part of the FOM "contract" among federates to interoperate at the specified levels. These specifications help to ensure a common perception of the simulation space across federates in a federation, lowering the potential for inconsistency.

4.4.2 Table format

The attribute table of a FOM shall describe all object attributes represented in a federation. The attribute table of a SOM shall describe all class attributes that are published or subscribed to by the federate. The template that shall be used for the attribute table is provided by Table 7.

Table 7—Attribute table

Object	Attribute	Datatype	Update type	Update condition	D/A	P/S	Available dimensions	Transport-ation	Order
HLA object Root	HLA privilege ToDelete Object	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>
<object class>	<attribute>	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>
	<attribute>	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>
	<attribute>	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>
<object class>	<attribute>	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>
	<attribute>	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>
	<attribute>	<datatype>	<update type>	<update condition>	<d/a>	<p/s>	<dimensions>	<transport>	<order>

The first column (Object) shall contain names from the object class structure table for object classes that are associated with attributes. Object class names shall include the parentage (superclasses) of the class to the depth necessary to uniquely identify the class in this table and may include the full parentage of the class. In general, to reduce redundancy, attributes should be specified for classes at the highest point in the hierarchy to which they apply. For example, if all air vehicles have an attribute of *MinimumTurnRadiusAt MaximumSpeed*, some redundancy can be avoided if this attribute is specified just once for the entire class of *AirVehicle*. Given that all object classes inherit the attribute types of their superclasses, the subclasses of *AirVehicle*, such as *FixedWing* and *RotaryWing,* also have this attribute with its specified characteristics.

The second column (Attribute) shall list the attributes of the specified object class. Attribute names shall adhere to HLA naming conventions (see 3.3.1). The names assigned to attributes of any particular object class shall not duplicate (overload) the names of attributes of this class or any higher level superclass. There may be many attributes for a single object class.

The third column (Datatype) shall identify the datatype of the attribute. This datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in subclauses of 4.12. A value of "NA" may also be provided if the attribute never contains a value; an example of such an attribute is a token that is owned but never updated.

Although an attribute's datatype can be specified "NA," that attribute must still have valid (non-"NA") transportation and order types specified. When a datatype of "NA" is used, the update type, update condition, and available dimensions shall also be "NA."

The fourth column (Update type) shall record the policy for updating an instance of the class attribute. The unique designations for this column shall be as follows:

— *Static*: The value of the attribute is static; the federate updates it initially and when requested.
— *Periodic*: The federate updates the attribute at regular time intervals.
— *Conditional*: The federate updates the attribute when unique conditions dictate.
— *NA*: The federate never provides a value for this attribute.

The fifth column (Update condition) shall expand and explain the policies for updating an instance of the class attribute. When the update type is *periodic*, a rate of the number of updates per time unit shall be specified in the Update Condition column. Attributes with a *conditional* update type shall have the conditions for update specified in the "Update condition" column. The "Update condition" column may also be used to specify initial conditions for instance attribute updates. If a federate is capable of varying the rate and circumstances of the update of an instance object attribute, this information shall also be recorded in this column or through the OMT notes feature. If the update type is *Static* or *NA*, "NA" shall be entered in this column.

The sixth column (D/A) shall indicate whether ownership of an instance of the class attribute can be divested or acquired.

In a FOM, if an instance attribute can be divested by a federate, it should be acquirable by some other federate in the federation. Therefore, the unique designations for this column shall be as follows:

— *N (NoTransfer)*: Ownership of an instance of this class attribute is not transferable in this federation.

— *DA (DivestAcquire)*: Some federate is capable of divesting ownership of instances of this class attribute to another federate and another federate is capable of acquiring ownership of instances of this class attribute.

In a SOM, the federate may be able to divest ownership of an instance attribute, acquire ownership, both, or neither. Moreover, the corresponding class attribute shall be published for the federate to divest or acquire ownership. The unique designations for this column shall be as follows:

— *D (Divest)*: The federate is capable of publishing the class attribute and can divest ownership of an instance of this attribute to another federate using the HLA ownership management services.

— *A (Acquire)*: The federate is capable of publishing the class attribute and can acquire ownership of an instance of this attribute from another federate.

— *N (NoTransfer)*: The federate is neither capable of divesting ownership of instances of this class attribute nor acquiring ownership of instances of this attribute.

— *DA (DivestAcquire)*: The federate is capable of both divesting ownership of instances of this class attribute and acquiring ownership of instances of this class attribute.

The seventh column (P/S) shall identify the capabilities of a federate or federation with respect to class attribute publishing and subscribing.

In a SOM, the entry for this column shall take one of the following values:

— *P (Publish)*: The federate is capable of publishing this class attribute, either in the context of the class where it is declared or in at least one subclass of that class (i.e., capable of invoking appropriate HLA publishing services with the class as a supplied argument).

— *S (Subscribe)*: The federate is capable of subscribing to this class attribute, either in the context of the class where it is declared or in at least one subclass of that class (i.e., capable of invoking HLA appropriate subscribing services with the class as a supplied argument).

— *PS (PublishSubscribe)*: The federate is capable of both publishing and subscribing to this class attribute.

— *N (Neither)*: The federate neither publishes nor subscribes to this class attribute.

In a FOM, the same basic designations shall apply.

The eighth column (Available dimensions) shall record the association of the class attribute with a set of dimensions if a federate or federation is using DDM services for this attribute. The column shall contain a comma-separated list of names of rows in the dimension table described in 4.6. Class attributes can be individually associated with dimensions, regardless of the dimensions associated with other attributes of the same class. For SOMs and FOMs of federates and federations, respectively, that are not using DDM services or for class attributes that are not associated with dimensions, “NA” shall be entered in this column.

The ninth column (Transportation) shall specify the type of transportation to be used with this attribute. These values shall be chosen from the name of any row in the transportation type table as described in 4.10.

The tenth column (Order) shall specify the order of delivery to be used with instances of this class attribute. Valid values for entries in this column are as follows:

— *Receive*: Instances of the class attribute are delivered to a receiving federate in an undetermined order.

— *TimeStamp*: Instances of the class attribute are delivered to a receiving federate in an order determined by time stamps assigned when the instance attributes were updated.

NOTE—One entry is required for Table 7. For this entry, the first column shall contain “HLAobjectRoot” and the second shall contain “HLAprivilegeToDeleteObject;” subsequent columns shall be completed as appropriate for the federate or federation. Uses and restrictions on this attribute are described in IEEE Std 1516.1-2000.

4.4.3 Inclusion criteria

All instance attributes whose values can be exchanged during the course of an HLA federation execution shall be documented in the attribute table of a FOM. All class attributes that can be either published or subscribed to by an individual federate shall be documented in the attribute table of its SOM.

4.4.4 Example

Table 8 shows examples of attributes from the restaurant application described in 4.2.4. First, the *Employee* object class is characterized according to the four attributes shown in the table. The datatypes specified are defined in the examples provided in 4.12. Each of these four attributes is updated conditionally except for the *YearsOfService* attribute, which is updated periodically (yearly) on the employee’s start date anniversary. As with most of the domain-specific attributes shown in this example, the attributes of *Employee* are assumed divestable, acquirable, publishable, and subscribable.

The *Waiter* subclass of *Employee* is shown with three attributes. These are in addition to the four inherited attributes from its superclass. Each of the first two attributes, *Efficiency* and *Cheerfulness*, is intended to represent a performance measure that is assigned to the employee at performance reviews. The third attribute is intended to represent the state of the employee (the task he/she is performing) at any point in time during restaurant operations.

In the final two entries, the *Drink* and *Soda* classes are each shown with a single attribute. The *NumberCups* attribute of the *Drink* class has been associated with the *BarQuantity* dimension, and the *Flavor* attribute of the *Soda* class has been associated with the *SodaFlavor* and *BarQuantity* dimensions; these dimensions are shown in the dimension table example in 4.6.4.

The HLAprivilegeToDeleteObject attribute of class HLAobjectRoot is a different kind of attribute than the others in the table. This attribute has a datatype of “NA,” meaning that an instance of this attribute does not have a value to be updated. Instead, it is more of a token attribute, in which information is conveyed simply by determining which, if any, federate owns the attribute at any time. In this case, HLAprivilegeToDeleteObject is a token attribute known by the RTI and used to determine whether or not a federate is permitted to delete an object instance.

Two examples of notes have also been included in Table 8; this feature is fully explained in 4.13.

Table 8—Attribute table example

Object	Attribute	Datatype	Update type	Update condition	D/A	P/S	Available dimensions	Transport-ation	Order
HLAobject Root	HLA privilege ToDelete Object	NA	NA	NA	N	N	NA	HLAreliable	Time Stamp
Employee	PayRate	DollarRate	Conditional	Merit increase *[1,2]	DA	PS	NA	HLAreliable	Time Stamp
	YearsOf Service	Years	Periodic	1/year *[2]	DA	PS	NA	HLAreliable	Time Stamp
	Home Number	HLAASCII string	Conditional	Employee request	DA	PS	NA	HLAreliable	Time Stamp
	Home Address	Address Type	Conditional	Employee request	DA	PS	NA	HLAreliable	Time Stamp
Employee. Waiter	Efficiency	Waiter Value	Conditional	Performance review	DA	PS	NA	HLAreliable	Time Stamp
	Cheerful-ness	Waiter Value	Conditional	Performance review	DA	PS	NA	HLAreliable	Time Stamp
	State	Waiter Tasks	Conditional	Work flow	DA	PS	NA	HLAreliable	Time Stamp
Food.Drink	Number Cups	DrinkCount	Conditional	Customer request	N	PS	BarQuantity	HLAreliable	Time Stamp
Food.Drink. Soda	Flavor	FlavorType	Conditional	Customer request	N	PS	SodaFlavor, BarQuantity	HLAreliable	Time Stamp

4.5 Parameter table

4.5.1 Purpose/background

Most interaction classes are characterized according to a list of one or more interaction parameters. Interaction parameters are used to associate relevant and useful information with classes of interactions. Examples of interaction parameters include object class names, object instance attribute values, strings, and numerical constants. Although some circumstances may require that a subset of the associated parameters be sent with an interaction, other circumstances will require that the full set of parameters be sent. However, for every interaction class identified in the interaction class structure table, the full set of parameters associated with that interaction class shall be described in the parameter table. The interaction class "HLAinteractionRoot" shall be a superclass of all other interaction classes in a FOM or SOM: Parameters may be assigned to it like any other interaction class.

Unlike class attributes, interaction parameters may not be subscribed to on an individual basis. Thus, dimension, transportation, and delivery order information is specified at the interaction class level rather than at the parameter level.

Interaction parameters may be associated with an interaction class at any level of an interaction class hierarchy. An interaction class inherits the parameters defined for its superclasses. In fact, the mechanisms and rules for inheritance of interaction parameters are identical to those of attributes. Thus, the specific placement of parameters throughout an interaction class hierarchy for a given federation is driven by the same needs that drive the placement of attributes in an object class hierarchy.

4.5.2 Table format

The parameter table of an HLA object model is designed to provide descriptive information about all parameters of all interactions represented in a federation. The template that shall be used for the parameter table is provided in Table 9.

Table 9—Parameter table

Interaction	Parameter	Datatype	Available dimensions	Transportation	Order
<interaction class>	<parameter>	<datatype>	<dimensions>	<transportation>	<order>
	<parameter>	<datatype>			
	<parameter>	<datatype>			
<interaction class>	<parameter>	<datatype>	<dimensions>	<transportation>	<order>
	<parameter>	<datatype>			
	<parameter>	<datatype>			

The first column (Interaction) shall contain names of interaction classes from the interaction class structure table. Interaction class names shall include the parentage (superclasses) of the class to the depth necessary to uniquely identify the class in this table and may include the full parentage of the class. In general, to reduce redundancy, parameters should be specified for classes at the highest point in the hierarchy in which they represent useful information, although this is not required. For example, if all weapon firings include a parameter that defines the infrared signature of the platform at the time the firing occurs, some redundancy will be avoided if this parameter is specified just once at the uppermost level of a *WeaponFires* class. Given that all interaction subclasses inherit the parameter types of their superclasses, the subclasses of *WeaponFires*, such as *TankFires* and *TELfires,* also have this parameter with its specified characteristics. Note that an interaction class need not contain any parameters; however, all interaction classes shall be included in this table so that transportation and order can be specified.

The second column (Parameter) shall list the parameters of each interaction. Parameter names shall adhere to HLA naming conventions (see 3.3.1). The names assigned to parameters of any particular interaction class shall not duplicate (overload) the names of parameters of this class or any higher level superclass. There may be many parameters for a single interaction class. If no parameters are specified for an interaction class, "NA" shall be entered in this column.

The third column (Datatype) shall identify the datatype of the parameter. This datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in 4.12. If the interaction class has no parameters, "NA" shall be entered for this column.

The fourth column (Available dimensions) shall record the association of an interaction class with a set of dimensions if the federate or federation is using DDM services for this interaction. The column shall contain a comma-separated list of names of rows from the dimension table described in 4.6. This indicates that whole interactions of this class will be subject to filtering through regions containing a subset of these dimensions when the interaction is sent. The available dimensions of an interaction class shall be a superset of the available dimensions of its immediate superclass. For SOMs and FOMs in which DDM services are not used or for interactions that are not associated with dimensions, "NA" shall be entered in this column.

The fifth column (Transportation) shall specify the type of transportation to be used with this interaction. These values shall be chosen from the name of any row in the transportation type table as described in 4.10.

The sixth column (Order) shall specify the order of delivery to be used with this interaction. Valid values for entry in this column are as follows:

— *Receive*: The interaction is delivered to a receiving federate in an undetermined order.
— *TimeStamp*: The interaction is delivered to a receiving federate in an order determined by a time stamp assigned when the interaction was initiated.

4.5.3 Inclusion criteria

In federations, any information elements that can be provided and associated with a given interaction class (by the class publishers) that are deemed to be useful to subscribers to that interaction class shall be included and documented as interaction parameters in the parameter table. For individual federates, SOM developers should associate with their publishable interaction classes whatever information they feel will be needed by subscribers to their interactions to calculate changes to the values of affected instance attributes. In addition, for interaction classes that the federate can subscribe to, SOM developers should determine what types of information need to be included with those interactions so that the federate can calculate associated effects.

4.5.4 Example

Table 10 shows a partial example of interactions and parameters chosen from the restaurant application described in 4.2.4. Here, the *FoodServed.MainCourseServed* interaction has three parameters associated with it. The third parameter uses a Boolean datatype to reflect whether the meal was served in a reasonable amount of time. In addition, this interaction has been associated with the *WaiterId* dimension from the dimension table example in 4.6.4 as a means of filtering on waiter identification numbers. *CustomerSeated* is an example of an interaction with no parameters; the sending and the receiving of the interaction is sufficient to convey the necessary information.

Table 10—Parameter table example

Interaction	Parameter	Datatype	Available dimensions	Transportation	Order
CustomerSeated	NA	NA	NA	HLAreliable	TimeStamp
FoodServed.MainCourse Served	TemperatureOk	ServiceStat	WaiterId	HLAreliable	TimeStamp
	AccuracyOk	ServiceStat			
	TimelinessOk	HLAboolean			

4.6 Dimension table

4.6.1 Purpose/background

Federations use DM services to enable the flow of instance attribute and interaction data between federates and to limit the delivery of some data on the basis of object class, interaction class, and attribute name. This reduction of data may be insufficient to meet the needs of federations with large numbers of federates, large numbers of objects or interactions in classes, or large numbers of instance attribute updates per object. Such federations can use DDM services to further reduce the volume of data delivered to federates. When DDM services are used by a federation, a common framework for specifying the data distribution model for the federation is essential for the following reasons:

— It provides a commonly understood mechanism for specifying the exchange of public data and DDM coordination among members of a federation.
— It facilitates the design and application of common tool sets for specification of federation DDM needs.

Dimensions are the most fundamental DDM concept. Each attribute or interaction with which DDM services can be used will have a set of available dimensions, as indicated in the attribute table or parameter table, respectively. Each set of available dimensions is a subset of all dimensions that are defined in the dimension table, and defines a multidimensional coordinate system through which federates either express an interest in receiving data or declare their intention to send data. These intentions are specified by creating regions within the coordinate system that indicate particular areas of interest. The regions associated with an instance attribute or interaction shall contain only dimensions that are a subset of the attribute's or interaction's available dimensions, but need not contain all of the available dimensions. These regions are then used in one of the two following ways:

— *Subscription regions*: Sets of ranges that narrow the scope of interest of the subscribing federate.

— *Update regions*: Sets of ranges that are guaranteed to enclose an object's location in the coordinate system.

During development of an HLA federation that uses DDM, it is critical that all federation members achieve a common understanding of DDM dimensions and their semantics, and that they agree to a common set of dimension specifications. These agreements are necessary for federates to filter instance attribute updates and interactions in a correct and it consistent manner. This shall include the names of dimensions, the association of attributes and interactions with dimensions, the federate view of each dimension, and the default ranges the RTI should supply for available dimensions of attributes and interactions when a federate does not specify ranges at region creation time.

The federate view of each dimension is different from the RTI view of each dimension. The federate view gives each dimension user-defined semantics and an associated datatype in which values are expressed. The federate shall provide a normalization function for each dimension, that maps values from the federate view of a dimension to values in the RTI view of a dimension. Note that the federate view of a dimension is the same for each federate in a federation.

The RTI view of a dimension is an interval of nonnegative integers. The lower bound is fixed for all dimensions, but the upper bound varies by dimension as indicated in the dimension table. The resolution of the RTI view of the dimension does not have to be the same as the resolution of the federate view of the same dimension.

The dimensions form the basis by which update and subscription regions are specified by the federates to the RTI. By having this agreement about the meaning of dimensions enforced at the federate level, the RTI can calculate the overlaps of update and subscription regions efficiently without having to understand the semantics of the dimensions. The dimension table shall record all of the elements necessary to specify this agreement in a standard format.

4.6.2 Table format

The template that shall be used for recording dimensions is illustrated in Table 11.

Table 11—Dimension table

Name	Datatype	Dimension upper bound	Normalization function	Value when unspecified
<dimension>	<type>	<bound>	<normalization function>	<default range/excluded>
<dimension>	<type>	<bound>	<normalization function>	<default range/excluded>
<dimension>	<type>	<bound>	<normalization function>	<default range/excluded>

The first column (Name) shall specify the name of the dimension. Dimension names shall adhere to HLA naming conventions (see 3.3.1). The names of dimensions in this column shall be unique. Each dimension represents a specific characteristic that may be used for filtering. For example, a dimension named *AltitudeLimits* might indicate the desire to filter information based on altitude.

The second column (Datatype) shall identify the datatype for the federate view of the dimension. This datatype shall be chosen from the name of a simple datatype or an enumerated datatype, as described in 4.12.

The third column (Dimension upper bound) shall specify the upper bound for the dimension that meets the federation's requirement for dimension subrange resolution. The value for this column shall be a positive integer. This value limits the resolution the RTI must provide to meet the federation's filtering requirements. It is also the maximum value a federate may use when communicating a range in this dimension to the RTI. For example, if a dimension represents radio frequencies in the range 88.1 MHz to 107.3 MHz in increments of .2, a dimension upper bound of 97 is sufficient.

The fourth column (Normalization function) shall specify the map from a subscription/update region's bounding coordinates to nonnegative integer subranges in the range [0, dimension upper bound). Because regions communicated with an RTI are expressed by bounding coordinates expressed in values from an RTI's view of dimensions, normalization functions shall be used by federates to construct regions. For each dimension in a region, that dimension's normalization function shall be used to map the dimension bounds as expressed in the federate view of the dimension to subranges of [0, dimension upper bound). To ensure that the federation exhibits correct semantics, federation participants should agree on the normalization functions that they will use for each dimension. This agreement shall include the specification of the normalization function and all values that are to be used by the function. If an attribute or parameter is used in the normalization function, its class hierarchy shall be specified in dot notation to the extent necessary to identify the attribute or parameter uniquely. The normalization function shall specify whether it uses a subset of the entire range or a subset of the datatype that is the basis of the dimension type. Example normalization functions are listed in Annex B.

The fifth column (Value when unspecified) shall specify a default range for the dimension that the RTI is to use in overlap calculations if the dimension is an available dimension of an attribute or interaction and has been left unspecified when a federate creates a region that is subsequently used (either for update or for subscription) with the attribute or interaction. The default range is specified as a nonnegative integer subrange of [0, dimension upper bound). Ranges shall be specified either by two values separated by two periods or by a single value. All ranges are closed on the lower bound (indicated by a left bracket), and open on the upper bound (indicated by a right parenthesis). The use of a single value shall specify a "point" range with the specified point as the lower bound and specified point plus one (1) for the upper bound, so that no other valid value lies between these points. An entry of "Excluded" indicates that the dimension shall not be used in overlap calculations unless a specific range is provided in the region specification.

4.6.3 Inclusion criteria

Federations that require more data reduction than the class-based filtering provided by DM services should use the dimension table to document federation-wide agreements on the semantics and use of DDM dimensions. The type and normalization function information is guidance to federates on the consistent use of dimensions to maintain correct DDM semantics across the federation. Similarly, federates that normally operate in such federations should document the mechanisms that they use to take advantage of DDM.

4.6.4 Example

Table 12 illustrates an example of the use of the dimension table. The first dimension, *SodaFlavor*, is an enumerated dimension, and its *linearEnumerated* normalization function (see Annex B) would map *Cola* to [0 .. 1), *Orange* to [1 .. 2), and *RootBeer* to [2 .. 3). The next two dimensions, *BarQuantity* and *WaiterId*, are integer-based dimensions and would be mapped at uniform intervals across the ranges [0 .. 25) and [0 .. 20), respectively. Notice that the default *Value When Unspecified* for *BarQuantity* is a point range that is equivalent to [0 .. 1).

Table 12—Dimension table example

Name	Datatype	Dimension upper bound	Normalization function	Value when unspecified
SodaFlavor	FlavorType	3	linearEnumerated (Flavor, [Cola, Orange, RootBeer])	[0 .. 3)
BarQuantity	DrinkCount	25	linear (NumberCups, 1, 25)	[0)
WaiterId	EmplId	20	linear (WaiterId, 1, 20)	Excluded

4.7 Time representation table

4.7.1 Purpose/background

Simulations provide a means of exercising system models over time. During a federation execution, time may play two roles. Federates may associate themselves with points on the HLA time axis, and they may associate some of their activities, such as updating an instance attribute's value or sending an interaction, with points on the HLA time axis. These points on the HLA time axis are referred to as time stamps. As a federation execution progresses, federates may advance along the HLA time axis.

The specific strategy used to advance time in a simulation is driven by the simulation's purpose. For example, faster-than-real time, event-stepped simulations are frequently used in analysis applications, whereas real time, time-stepped simulations are often used in training applications. The strategy chosen for advancing time is an important consideration in the design of a simulation, because it affects both simulation performance and the ability of a simulation to interoperate with other simulations in a federation. For this reason, it is important to document how a simulation supports the advancement of time in a SOM.

IEEE Std 1516.1-2000 provides a variety of time management services to control the advancement of federates along the HLA time axis. These services allow multiple federates with differing internal time advancement strategies to interoperate in a meaningful way. For this reason, it is vital for all federation members to agree on how time stamps are represented across the federation, and document this agreement in the federation's FOM.

During federation execution, time stamps are represented as instances of the time representation abstract datatype. This abstract datatype is provided to the RTI when a federate joins a federation execution so that an RTI will know how to use the time stamps. This table allows the datatype and semantics of the abstract datatype for time stamps to be explained in FOMs and SOMs.

It is also important to define the lookahead characteristics of both federates and federations. At the individual federate level, the specific means by which lookahead is supported may be a factor in assessing compatibility with other potential federates for a given federation application. For federations, it is important to document a common set of characteristics for how lookahead will be supported across the full federation. Thus, both SOMs and FOMs include representations of lookahead as well as time stamps. Representations of lookahead shall be drawn from nonnegative types.

An abstract datat ype for lookahead is supplied when a federate joins a federation execution. The lookahead entry in this table allows the datatype and semantics of this abstract datatype to be documented in FOMs and SOMs.

Semantic differences exist between the way time is represented for the purpose of depicting time stamps versus calculating lookahead. When depicting time stamps, time can be considered to be an absolute value on a timeline (the HLA time axis), and thus, time comparisons can be drawn to determine if one time stamp is greater than another. Lookahead, in contrast, represents a duration of time, which can be added to time stamps but is generally not used for comparison purposes.

4.7.2 Table format

The template that shall be used for recording time representation is illustrated in Table 13.

Table 13—Time representation table

Category	Datatype	Semantics
Time stamp	<type>	<semantics>
Lookahead	<type>	<semantics>

The first column (Category) presents the two time-related values that are to be specified in the table: time stamp and lookahead. *Time Stamp* and *Lookahead* correspond to the only two rows in this table.

The second column (Datatype) shall identify the datatype of the time value. This datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in 4.12. “NA” shall be entered in this column when time stamp or lookahead information is inappropriate to the federate or federation.

The third column (Semantics) shall expand and describe the use of the datatype for time values. “NA” shall be entered in this column when semantics information is unnecessary.

4.7.3 Inclusion criteria

All federations shall document their use of time stamp and lookahead via the time representation table in a FOM. If the federation (during the federation execution) does not use either time stamps or lookahead, “NA” shall be entered into the table. All federates shall document their representation of time stamps via the time representation table in a SOM to the extent that it is used in the federate. For federates that support lookahead, the representation of lookahead shall be documented via the time representation table in a SOM. For federates that have no temporal dimension, “NA” shall be entered in all columns of the time representation table in that federate’s SOM.

4.7.4 Example

Table 14 provides an example of how the OMT tables are used to describe the representation of time for the restaurant application.

Table 14—Time representation table example

Category	Datatype	Semantics
Time stamp	TimeType	Floating point value expressed in minutes
Lookahead	LAType	Floating point value expressed in minutes (non-negative)

4.8 User-supplied tag table

4.8.1 Purpose/background

The HLA RTI provides a mechanism for federates to supply tags with certain of the HLA services. These tags can be used to provide additional coordination and control over these services. The user-supplied tag table provides a means of documenting federation agreements regarding the datatype to be used with these tags.

4.8.2 Table format

The template that shall be used for recording information on user-supplied tags is illustrated in Table 15.

Table 15—User-supplied tag table

Category	Datatype	Semantics
Update/reflect	<type>	<semantics>
Send/receive	<type>	<semantics>
Delete/remove	<type>	<semantics>
Divestiture request	<type>	<semantics>
Divestiture completion	<type>	<semantics>
Acquisition request	<type>	<semantics>
Request update	<type>	<semantics>

The first column (Category) presents the HLA service categories that are capable of accepting a user-supplied tag

— *Update/Reflect*: Updating and the corresponding reflection of instance attribute values.

— *Send/Receive*: Sending and the corresponding reception of an interaction.

— *Delete/Remove*: Deletion and the corresponding removal of an object instance.

— *Divestiture Request:* Negotiations involved in divesting ownership of instance attributes.

— *Divestiture Completion*: Confirmation and completion of the negotiated divestiture of ownership of instance attributes.

— *Acquisition Request*: Negotiations involved in acquiring ownership of instance attributes.

— *Request Update*. Requesting update of instance attribute values and the corresponding RTI direction to provide instance attribute values.

These seven service categories correspond to the only seven rows in Table 15.

The second column (Datatype) shall identify the datatype of the user-supplied tag for those categories of service that the federate or federation designate as providing a user-supplied tag. This datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in sub-clauses of 4.12. "NA" shall be entered in this column when this category of user-supplied tag is not used by the federate or federation.

The third column (Semantics) shall expand and describe the use of the datatype for this user-supplied tag. "NA" shall be entered in this column when semantics information is unnecessary.

4.8.3 Inclusion criteria

All federations that make use of user-supplied tags shall document the appropriate representations of these tags in the user-supplied tag table in a FOM. All federates that can make use of user-supplied tags shall document their tag representations via the user-supplied tag table in a SOM.

4.8.4 Example

Table 16 provides an example of how the OMT tables are used to describe the representation of user-supplied tags by the restaurant federate. In this example, no tags are specified for update/reflect, send/receive, divestiture completion, or request update. A simple text string is used for delete/remove that describes the reasons for the action. A priority entry is specified for ownership transfer negotiations (both divestiture requests and acquisition requests). This entry indicates the criticality of the transfer; a high value tells the federate to accept or divest ownership immediately.

Table 16—User-supplied tag table example

Category	Datatype	Semantics
Update/reflect	NA	NA
Send/receive	NA	NA
Delete/remove	HLAASCIIstring	Reason for deletion
Divestiture request	PriorityLevel	High value for immediate transfer
Divestiture completion	NA	NA
Acquisition request	PriorityLevel	High value for immediate transfer
Request update	NA	NA

4.9 Synchronization table

4.9.1 Purpose/background

The HLA RTI provides a mechanism for federates to synchronize activities using a feature called Synchronization Points. The synchronization table provides the means for a federate to describe the synchronization points that it is capable of honoring and for a federation to document agreements regarding synchronization points to be used.

4.9.2 Table format

The template that shall be used for recording information to be used in synchronization points is illustrated in Table 17.

Table 17—Synchronization table

Label	Tag datatype	Capability	Semantics
<label>	<type>	<capability>	<semantics>
<label>	<type>	<capability>	<semantics>
<label>	<type>	<capability>	<semantics>

The first column (Label) shall contain a text string that defines the label associated with a synchronization point. Synchronization labels shall adhere to HLA naming conventions (see 3.3.1).

The second column (Tag datatype) shall identify the datatype of the user-supplied tag for those synchronization points that the federate or federation designate as providing a user-supplied tag. If used, this datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in 4.12.

The third column (Capability) shall indicate the level of interaction that a federate is capable of honoring. For FOMs, this column does not apply and shall contain “NA.” For SOMs, valid values for this column are as follows:

— *Register*: The federate is capable of invoking HLA services to register the synchronization point.

— *Achieve*: The federate is capable of invoking HLA services to indicate achieving the synchronization point.

— *RegisterAchieve*: The federate is capable of both registering and achieving the synchronization point.

— *NoSynch*: The federate is capable of neither registering nor achieving the synchronization point.

The fourth column (Semantics) shall expand and describe the use of the synchronization point.

4.9.3 Inclusion criteria

All federations that make use of synchronization points shall document the appropriate representations of these points in the synchronization table of a FOM. All federates that make use of synchronization points shall document them via the synchronization table in a SOM.

4.9.4 Example

Table 18 provides an example of how synchronization points might be described for a particular federate. In this example, no tags are specified for the first three synchronization points and a tag of type *TimeType* is included in the fourth. This federate is able to achieve all of the synchronization points and to register the fourth as well.

Table 18—Synchronization table example

Label	Tag datatype	Capability	Semantics
InitialPublish	NA	Achieve	Achieved when all classes are published and subscribed, and all initially present objects are registered
InitialUpdate	NA	Achieve	Achieved when instance attribute values for all initially present objects are updated
BeginTimeAdvance	NA	Achieve	Achieved when time management services are invoked
PauseExecution	TimeType	Register Achieve	Achieved when the time advance after the time in the user-supplied tag is attained; time advance requests should then cease

4.10 Transportation type table

4.10.1 Purpose/background

The HLA RTI provides different mechanisms for the transportation of data (interactions and object instance attribute values) among federates. The transportation type table provides a means for a federate designer to describe the types of transportation that can be supported and for federation designers to document agreements regarding transportation of instance attributes and interactions.

Two transportation types, HLAreliable and HLAbestEffort, are required by the HLA and are described in detail in IEEE Std 1516.1-2000. These types shall be provided by all RTIs. Other transportation types may be provided by specific RTIs.

4.10.2 Table format

The template that shall be used for recording information about transportation types is illustrated in Table 19. Note that the transportation types required by the HLA to be provided by all RTIs are predefined in the table. Other entries may be added after these entries.

Table 19—Transportation type table

Name	Description
HLAreliable	Provide reliable delivery of data in the sense that TCP/IP delivers its data reliably
HLAbestEffort	Make an effort to deliver data in the sense that UDP provides best-effort delivery
<name>	<description>

The first column (Name) shall contain a text string that defines the name associated with a transportation type. Transportation type names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Description) shall describe the transportation type.

4.10.3 Inclusion criteria

The two transportation types (HLAreliable and HLAbestEffort) that are required by the HLA shall be included in this table. Federations may choose to use one, both, or neither of these types depending on their needs. Other transportation types that are provided by specific RTIs shall be included in this table when used by a federate or federation.

4.10.4 Example

Table 20 provides an example of how the OMT tables are used to describe the transportation types. The two required transportation types are followed by a user-defined type that directs delivery with minimal latency.

Table 20—Transportation type table example

Name	Description
HLAreliable	Provide reliable delivery of data in the sense that TCP/IP delivers its data reliably
HLAbestEffort	Make an effort to deliver data in the sense that UDP provides best-effort delivery
LowLatency	Choose the delivery mechanism that results in the lowest latency from service initiation to callback invocation at the receiving federate

4.11 Switches table

4.11.1 Purpose/background

The HLA RTI performs actions on behalf of federates. Some of these actions may be enabled or disabled based on the capabilities of the federates or desires of the federation designers. These actions include automatically soliciting updates of instance attribute values when an object is newly discovered (controlled by the Auto Provide switch); and advising federates when certain events occur (controlled by the advisory switches). The switches table permits the specification of the initial setting of each switch, indicating whether the RTI should perform these actions. The functionality of these switches is described fully in IEEE Std 1516.1-2000.

Although the initial setting of each switch is specified in this table, the value of each switch may be changed during execution. The *Auto Provide* and *Convey Region Designator Sets* switches are controlled on a federation-wide basis; if any federate changes the switch settings, the changes affect how the RTI interacts with all federates in the execution. The advisory switches, however, are controlled on a per-federate basis. Each federate in a federation has the same initial settings, as indicated in this table, but changes that an individual federate makes to an advisory switch only affect how the RTI interacts with that federate. Likewise, changes made to the Service Reporting switch are made on a per-federate basis. The RTI initially either reports service invocations for all federates or for none, as indicated in this table; changes made during a federation execution only affect whether the RTI reports service invocations of a particular federate.

4.11.2 Table format

The switch settings shall be provided in a simple two-column table. The structure that shall be used to describe these settings is provided in Table 21.

Table 21—Switches table

Switch	Setting
Auto Provide	<auto provide>
Convey Region Designator Sets	<convey region designator sets>
Attribute Scope Advisory	<attribute scope advisory>
Attribute Relevance Advisory	<attribute relevance advisory>
Object Class Relevance Advisory	<object class relevance advisory>
Interaction Relevance Advisory	<interaction relevance advisory>
Service Reporting	<service reporting>

The first column (Switch) presents the name of the switch whose setting shall be provided in this table. The definitions of these switches are as follows:

— *Auto Provide*: Whether the RTI should automatically solicit updates from instance attribute owners when an object is discovered.

— *Convey Region Designator Sets*: Whether the RTI should provide the optional *Sent Region Set* argument with invocations of *Reflect Attribute Values* and *Receive Interaction*.

— *Attribute Scope Advisory*: Whether the RTI should advise federates when attributes of an object instance come into or go out of scope.

— *Attribute Relevance Advisory*: Whether the RTI should advise federates about whether they should provide attribute value updates for the value of an attribute of an object instance; the RTI bases this advisory on whether the value of the instance attribute is required by other federates.

— *Object Class Relevance Advisory*: Whether the RTI should advise federates about whether they should register instances of an object class; the RTI bases this advisory on whether other federates have expressed an interest in attribute(s) of the object class.

— *Interaction Relevance Advisory*: Whether the RTI should advise federates about whether they should send interactions of an interaction class; the RTI bases this advisory on whether other federates have expressed an interest in the interaction class.

— *Service Reporting*: Whether the RTI should report service invocations using MOM.

The second column (Setting) shall specify the setting for the switch. For SOMs, these entries are optional; "NA" shall be entered for all rows where no value is appropriate. For FOMs, entries shall be provided for all rows; valid values are as follows:

— *Enabled*: The switch is enabled, and the RTI should perform the action when appropriate.

— *Disabled*: The switch is disabled, and the RTI should not perform the action.

4.11.3 Inclusion criteria

The switch settings specified in this table shall be included in all FOMs to document agreements among federation developers as to the initial values of the switches at the beginning of federation execution. The switch settings may optionally be provided in SOMs to indicate switch settings that are desired by federate designers.

4.11.4 Example

Table 22 shows an example of the switches table for the Restaurant federate. In this example, the *Auto Provide* entry indicates that the federate developer does not desire the RTI to solicit updates of attributes of newly discovered object instances, the *Convey Region Designator Sets* entry indicates that the RTI should not provide the *Sent Region Set* argument, and the Service Reporting entry indicates that the RTI should not report service invocations using the MOM. However, the RTI should advise the federate regarding when attributes of object instances come into or go out of scope (*Attribute Scope Advisory*), whether the federate should register object instances based on subscriptions of other federates (*Object Class Relevance Advisory*), whether the federate should provide attribute value updates of object instances based on the interest of other federates (*Attribute Relevance Advisory*), and whether the federate should send interactions based on the interest of other federates (*Interaction Relevance Advisory*).

Table 22—Switches table example

Switch	Setting
Auto provide	Disabled
Convey region designator sets	Disabled
Attribute scope advisory	Enabled
Attribute relevance advisory	Enabled
Object class relevance advisory	Enabled
Interaction relevance advisory	Enabled
Service reporting	Disabled

4.12 Datatype tables

4.12.1 Purpose/background

Several of the OMT tables (attribute table, parameter table, dimension table, time representation table, user-supplied tag table, and synchronization table) provide columns for datatype specifications. This subclause describes additional tables that shall be used to specify the characteristics of these datatypes. A datatype used in these tables shall be specified in a row of one of the following tables: simple datatype table, enumerated datatype table, fixed record datatype table, array datatype table, or variant record datatype table. As described in the subclauses below, members of these datatype tables may contain members of other datatype tables in order to create arbitrarily complex datatypes.

NOTE—That the name of any datatype or data representation shall be unique among all datatypes and data representations.

4.12.2 Inclusion criteria

All federations shall describe and document the datatypes that are referenced in any OMT table that requires the specification of datatypes (attribute table, parameter table, dimension table, time representation table, user-supplied tag table, and synchronization table) in their FOM. They shall also describe and document datatypes referenced in other datatype tables. Similarly, federates shall document the datatypes referenced in these tables in their SOM.

4.12.3 Basic data representation table

Basic data representation is the underpinning of all OMT datatypes. Although it is not used as a datatype in any of the OMT tables, it forms the basis of the datatypes that are used there. Table 23 illustrates the format that shall be used for the basic representation of data. The table includes a predefined set of data representations (*HLAinteger16BE, HLAinteger32BE, HLAinteger64BE, HLAfloat32BE, HLAfloat64BE, HLAoctetPairBE, HLAinteger16LE, HLAinteger32LE, HLAinteger64LE, HLAfloat32LE, HLAfloat64LE, HLAoctetPairLE, HLAoctet*) that shall be present in all FOMs and SOMs; other entries may be added after these entries. Although these entries must be present, there is no requirement that federates or federations use or support all of these representations.

Table 23—Basic data representation table

Name	Size in bits	Interpretation	Endian	Encoding
HLAinteger16BE	16	Integer in the range $[-2^{15}, 2^{15} - 1]$.	Big	16-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger32BE	32	Integer in the range $[-2^{31}, 2^{31} - 1]$.	Big	32-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger64BE	64	Integer in the range $[-2^{63}, 2^{63} - 1]$.	Big	64-bit two's complement signed integer. The most significant bit contains the sign.
HLAfloat32BE	32	Single-precision floating-point number.	Big	32-bit IEEE normalized single-precision format (see IEEE Std. 754-1985).
HLAfloat64BE	64	Double-precision floating-point number.	Big	64-bit IEEE normalized double-precision format (see IEEE Std. 754-1985).
HLAoctetPairBE	16	16-bit value	Big	Assumed to be portable among hardware devices
HLAinteger16LE	16	Integer in the range $[-2^{15}, 2^{15} - 1]$.	Little	16-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger32LE	32	Integer in the range $[-2^{31}, 2^{31} - 1]$.	Little	32-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger64LE	64	Integer in the range $[- 2^{63}, 2^{63} - 1]$.	Little	64-bit two's complement signed integer. The most significant bit contains the sign.
HLAfloat32LE	32	Single-precision floating-point number.	Little	32-bit IEEE normalized single-precision format (see IEEE Std. 754-1985).
HLAfloat64LE	64	Double-precision floating-point number.	Little	64-bit IEEE normalized double-precision format (see IEEE Std. 754-1985).
HLAoctetPairLE	16	16-bit value	Little	Assumed to be portable among hardware devices
HLAoctet	8	8-bit value	Big	Assumed to be portable among hardware devices.
<name>	<size>	<interpretation>	<endian>	<encoding>
<name>	<size>	<interpretation>	<endian>	<encoding>

The first column (Name) shall identify the name of the basic data representation. Basic data representation names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Size in bits) shall define the size of the data representation in terms of the number of bits contained in the data representation. Entries in this column shall be consistent with the associated encoding descriptions.

The third column (Interpretation) shall describe how the data representation should be interpreted.

The fourth column (Endian) shall describe how multiple bytes within the representation are arranged. Valid values shall be as follows:

— *Big*: The most significant byte will come first.

— *Little*: The least significant byte will come first.

The fifth column (Encoding) shall describe, in detail, the encoding of the data representation (e.g., the bit ordering) as delivered to and received from the RTI.

An example of the use of the basic data representation table is provided in Table 24. This example includes the pre-defined set of basic data representations and follows with a user-defined entry (UnsignedShort).

Table 24—Basic data representation table example

Name	Size in bits	Interpretation	Endian	Encoding
HLAinteger16BE	16	Integer in the range $[-2^{15}, 2^{15}-1]$.	Big	16-bit two’s complement signed integer. The most significant bit contains the sign.
HLAinteger32BE	32	Integer in the range $[-2^{31}, 2^{31}-1]$.	Big	32-bit two’s complement signed integer. The most significant bit contains the sign.
HLAinteger64BE	64	Integer in the range $[-2^{63}, 2^{63}-1]$.	Big	64-bit two’s complement signed integer. The most significant bit contains the sign.
HLAfloat32BE	32	Single-precision floating-point number.	Big	32-bit IEEE normalized single-precision format (see IEEE Std. 754-1985).
HLAfloat64BE	64	Double-precision floating-point number.	Big	64-bit IEEE normalized double-precision format (see IEEE Std. 754-1985).
HLAoctetPairBE	16	16-bit value	Big	Assumed to be portable among hardware devices.
HLAinteger16LE	16	Integer in the range $[-2^{15}, 2^{15}-1]$.	Little	16-bit two’s complement signed integer. The most significant bit contains the sign.
HLAinteger32LE	32	Integer in the range $[-2^{31}, 2^{31}-1]$.	Little	32-bit two’s complement signed integer. The most significant bit contains the sign.
HLAinteger64LE	64	Integer in the range $[-2^{63}, 2^{63}-1]$.	Little	64-bit two’s complement signed integer. The most significant bit contains the sign.
HLAfloat32LE	32	Single-precision floating-point number.	Little	32-bit IEEE normalized single-precision format (see IEEE Std. 754-1985).

Table 24—Basic data representation table example *(continued)*

Name	Size in bits	Interpretation	Endian	Encoding
HLAfloat64LE	64	Double-precision floating-point number.	Little	64-bit IEEE normalized double-precision format (see IEEE Std. 754-1985).
HLAoctetPairLE	16	16-bit value	Little	Assumed to be portable among hardware devices.
HLAoctet	8	8-bit value	Big	Assumed to be portable among hardware devices.
UnsignedShort	16	Integer in the range $[0, 2^{16} – 1]$	Big	16-bit unsigned integer.

4.12.4 Simple datatype table

The simple datatype table shall be used to describe simple, scalar data items. Table 25 illustrates the format that shall be used for the simple datatype table. The table includes three predefined simple datatypes (*HLAASCIIchar, HLAunicodeChar, HLAbyte*) that shall be present in all FOMs and SOMs; additional entries may be added after them. Although these required datatypes must be present, no requirement exists that federates or federations use them.

Table 25—Simple datatype table

Name	Representation	Units	Resolution	Accuracy	Semantics
HLAASCIIchar	HLAoctet	NA	NA	NA	Standard ASCII character (see ANSI Std. X3.4-1986)
HLAunicodeChar	HLAoctetPairBE	NA	NA	NA	Unicode UTF-16 character (see *The Unicode Standard*, Version 3.0)
HLAbyte	HLAoctet	NA	NA	NA	Uninterpreted 8-bit byte
<simple type>	<representation>	<units>	<resolution>	<accuracy>	<semantics>
<simple type>	<representation>	<units>	<resolution>	<accuracy>	<semantics>

The first column (Name) shall identify the name of the simple datatype. Simple datatype names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Representation) shall identify the basic data representation of this datatype. It shall be the name of a row in the basic data representation table.

The third column (Units) shall identify the units of measure (e.g., m, km, kg) for the datatype whenever such units exist. Any units entered in this column also specify the units of the entries in the Resolution and Accuracy columns that follow it. “NA” shall be entered in this column for datatypes without units.

The fourth column (Resolution) shall describe the precision of measure for the datatype. In general, this entry specifies the smallest resolvable increment between different values that can be effectively discriminated. In some situations, such as when values are stored in floating point datatypes, the resolution might vary with the magnitude of the value. Hence, in cases like this, a better sense of the resolution may be conveyed by the datatype. “NA” shall be entered in this column for datatypes for which resolution information does not apply.

The fifth column (Accuracy) shall describe maximum deviation of the value from its intended value in the federate or federation. This is ordinarily expressed as a dimensioned value, but it may also be *perfect* for discrete values. “NA” shall be entered in this column for datatypes for which accuracy information does not apply.

The sixth column (Semantics) shall describe the meaning and use of the datatype. “NA” shall be entered in this column when semantics information is unnecessary.

An example of the use of the simple datatype table is provided in Table 26. These datatypes are used in examples elsewhere in this document. Note that *DrinkCount* is used for attributes and for dimension definition, and *EmplId* is used for dimension and array definition.

Table 26—Simple datatype table example

Name	Representation	Units	Resolution	Accuracy	Semantics
HLAASCIIchar	HLAoctet	NA	NA	NA	Standard ASCII character (see ANSI Std. X3.4-1986)
HLAunicodeChar	HLAoctetPairBE	NA	NA	NA	Unicode UTF-16 character (see *The Unicode Standard*, Version 3.0)
HLAbyte	HLAoctet	NA	NA	NA	Uninterpreted 8-bit byte
TimeType	HLAfloat32BE	Minutes	0.01667	NA	Time representation
LAType	HLAfloat32BE	Minutes	0.01667	NA	Time interval (nonnegative)
DollarRate	HLAfloat32BE	$/hour	0.01	Perfect	Cost per hour
Years	HLAinteger32BE	Years	1	Perfect	Elapsed time in years
DrinkCount	UnsignedShort	Cups	1	Perfect	Measure of number of drinks
EmplId	HLAinteger32BE	NA	1	Perfect	Employee identifier
RateScale	HLAinteger32BE	NA	1	Perfect	Evaluation of staff where 1 = best

4.12.5 Enumerated datatype table

The enumerated datatype table shall be used to describe data elements that can take on a finite discrete set of possible values. Table 27 illustrates the format that shall be used for the enumerated datatype table. The table includes a single predefined datatype (*HLAboolean*) that shall be present in all FOMs and SOMs; additional entries may be added after it. Although this required datatype must be present, no requirement exists that federates or federations use it.

The first column (Name) shall identify the name of the enumerated datatype. Enumerated datatype names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Representation) shall identify the basic data representation that forms the basis of this datatype. It shall be the name of a row in the basic data representation table that specifies discrete values.

The third column (Enumerator) shall provide the names of all enumerators associated with this datatype. Enumerator names shall adhere to HLA naming conventions (see 3.3.1).

The fourth column (Values) shall provide values that correspond to each enumerator in the same row. The values presented in the enumerator shall adhere to the data representation specified in the Representation column. If multiple values are associated with a single enumerator, commas shall be used to separate the values.

The fifth column (Semantics) shall describe the meaning and use of this datatype. “NA” shall be entered in this column when semantics information is unnecessary.

Table 27—Enumerated datatype table

Name	Representation	Enumerator	Values	Semantics
HLAboolean	HLAinteger32BE	HLAfalse	0	Standard Boolean type
		HLAtrue	1	
<enumerated type>	<representation>	<enumerator 1>	<value(s)>	<semantics>
		…	…	
		<enumerator n>	<value(s)>	
<enumerated type>	<representation>	<enumerator 1>	<value(s)>	<semantics>
		…	…	
		<enumerator m>	<value(s)>	

An example of the use of the enumerated datatype table is provided in Table 28.

Table 28—Enumerated datatype table example

Name	Representation	Enumerator	Values	Semantics
HLAboolean	HLAinteger32BE	HLAfalse	0	Standard Boolean type
		HLAtrue	1	
PriorityLevel	HLAinteger32BE	Low	0	General three-level priority indicator
		Medium	1	
		High	2	
WaiterTasks	HLAinteger32BE	TakingOrder	1	Possible activities of waiters
		Serving	2	
		Cleaning	3	
		CalculatingBill	4	
		Other	5	
FlavorType	HLAinteger32BE	Cola	101	Possible flavors of soda
		Orange	102	
		RootBeer	103	
		Cream	104	
ExperienceLevel	HLAinteger32BE	Trainee	0	Level of experience of waiters
		Apprentice	1	
		Journeyman	2	
		Senior	3	
		Temporary	4	
		Master	5	

4.12.6 Array datatype table

The array datatype table shall describe indexed homogenous collections of datatypes; these constructs are also known as arrays or sequences. Table 29 illustrates the format that shall be used for the array datatype table. The table includes three predefined array datatypes (*HLAASCIIstring, HLAunicodeString, HLAopaqueData*) that shall be present in all FOMs and SOMs; Additional entries may be added after them. Although these required datatypes must be present, no requirement exists that federates or federations use them.

Table 29—Array datatype table

Name	Element type	Cardinality	Encoding	Semantics
HLAASCIIstring	HLAASCIIchar	Dynamic	HLAvariableArray	ASCII string representation
HLAunicodeString	HLAunicodeChar	Dynamic	HLAvariableArray	Unicode string representation
HLAopaqueData	HLAbyte	Dynamic	HLAvariableArray	Uninterpreted sequence of bytes
<array type>	<type>	<cardinality>	<encoding>	<semantics>
<array type>	<type>	<cardinality>	<encoding>	<semantics>

The first column (Name) shall identify the name of the array datatype. Array datatype names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Element Type) shall identify the datatype of an element of this datatype. Values in this column shall be names from other datatype tables (simple datatype table, enumerated datatype table, fixed record datatype table, array datatype table, or variant record datatype table). Note that higher dimensional arrays may be specified by using array datatypes in the element type column of this table.

The third column (Cardinality) shall contain the number of elements that are contained in the array datatype. Multidimensional arrays can be specified via a comma-separated list of values, each representing one dimension. If the number of elements in the array varies during the use of the datatype, a range of values may be provided; if used, this range shall take the form of upper and lower bound values, separated by two periods and surrounded by brackets. Alternatively, the keyword “Dynamic” may be entered into this column for these types of arrays.

The fourth column (Encoding) shall describe, in detail, the encoding of the array datatype (e.g., the sequence of elements and the order of elements in multi-dimensional arrays) as delivered to and received from the RTI. Encoding of elements in an array is determined by the datatype specified in the Element type column. Encoding of the array is determined by the encoding specified in the Encoding column. Array data should be interpreted first through the array encoding scheme, and then through the encoding scheme of the element’s datatype.

For one-dimensional array datatypes that use one of the two predefined array encodings, the appropriate keyword “HLAfixedArray” or “HLAvariableArray” shall be entered in the Encoding column. The “HLAfixedArray” encoding is intended for arrays with fixed cardinality and consists of the encoding of each element in sequence. The “HLAvariableArray” encoding is intended for arrays with variable (including dynamic) cardinality, and consists of the number of elements encoded as an *HLAinteger32BE*, followed by the encoding of each element in sequence. Full details of these predefined encodings are given in 4.12.9.

The fifth column (Semantics) shall describe the meaning and use of this datatype. “NA” shall be entered in this column when semantics information is unnecessary.

An example of the use of the array datatype table is provided in Table 30. *Employees* is a fixed-length array type that holds the employee identifiers of everyone currently working; It uses the standard *HLAfixedArray* encoding. *AddressBook* is an array type that can be used to hold a collection of addresses; because it is encoded as a sparse array, addresses can be updated selectively without sending every element of the array. Because one may want to update addresses selectively without sending every element of the array, a special encoding is defined for this datatype.

Table 30—Array datatype table example

Name	Element type	Cardinality	Encoding	Semantics
HLAASCIIString	HLAASCIIchar	Dynamic	HLAvariableArray	ASCII string representation
HLAunicodeString	HLAunicodeChar	Dynamic	HLAvariableArray	Unicode string represenation
HLAopaqueData	HLAbyte	Dynamic	HLAvariableArray	Uninterpreted sequence of bytes
Employees	EmplId	10	HLAfixedArray	Identifiers of employees currently working
AddressBook	AddressType	Dynamic	An HLAinteger32BE followed by a set of index-value tuples. Each tuple consists of an HLAinteger32BE indicating the array index, followed by the element for that index. The initial HLAinteger32BE indicates the number of index-value pairs to follow, since all array elements need not be included.	Collection of all employee addresses

4.12.7 Fixed record datatype table

The fixed record datatype table shall be used to describe heterogeneous collections of types; these constructs are also known as records or structures. Each entry in the fixed record datatype table may contain fields that are of other types, such as simple datatypes, fixed records, arrays, enumerations, or variant records. This allows users to build “structures of data structures” according to the needs of their federate or federation. Table 31 illustrates the format that shall be used for the fixed record datatype table.

Table 31—Fixed record datatype table

Record name	Field			Encoding	Semantics
	Name	Type	Semantics		
<record type>	<field 1>	<type 1>	<semantics>	<encoding>	<semantics>
	…	…	…		
	<field n>	<type n>	<semantics>		
<record type>	<field 1>	<type 1>	<semantics>	<encoding>	<semantics>
	…	…	…		
	<field m>	<type m>	<semantics>		

The first column (Record name) shall identify the name of the fixed record datatype. Record names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Field name) shall identify the name of a field in the fixed record datatype. Field names shall adhere to HLA naming conventions (see 3.3.1).

The third column (Field type) shall identify the datatype of the field. Values in this column shall be names from other datatype tables (simple datatype table, enumerated datatype table, fixed record datatype table, array datatype table, or variant record datatype table).

The fourth column (Field semantics) shall describe the meaning and use of this field. "NA" shall be entered in this column when semantics information is unnecessary.

The fifth column (Encoding) shall describe, in detail, the encoding of the fixed record datatype (i.e., the organization of fields) as delivered to and received from the RTI. Encoding of fields in a fixed record is determined by the datatype specified in the Field Type column. Encoding of the fixed record is determined by the encoding specified in the Encoding column. Record data should be interpreted first through the record encoding scheme, and then through the encoding scheme of the field's datatype.

For fixed record datatypes that use the predefined fixed record encoding, the keyword "HLAfixedRecord" shall be entered in the Encoding column (see 4.12.9). The "HLAfixedRecord" encoding consists of the encoding of each field in sequence, in the order in which they are declared. Full details of this predefined encoding are given in 4.12.9.

The sixth column (Semantics) shall describe the meaning and use of this datatype. "NA" shall be entered in this column when semantics information is unnecessary.

An example of the use of the fixed record datatype table is provided in Table 32. These datatypes are used in examples elsewhere in this document. *ServiceStat* combines three *HLAboolean* values into one record. *AddressType* creates a record of components of a mailing address. Both datatypes use the standard *HLA-fixedRecord* encoding.

Table 32—Fixed record datatype table example

Record name	Field			Encoding	Semantics
	Name	Type	Semantics		
ServiceStat	EntreeOk	HLAboolean	Entree status	HLAfixedRecord	Check-off on whether the server performed properly on elements of the meal
	Veggy1Ok	HLAboolean	Vegetable 1 status		
	Veggy2Ok	HLAboolean	Vegetable 2 status		
AddressType	Name	HLAASCIIstring	Employee name	HLAfixedRecord	Mailing address
	Street	HLAASCIIstring	Street address		
	City	HLAASCIIstring	City name		
	State	HLAASCIIstring	State abbreviation		
	Zip	HLAASCIIstring	Postal code		

4.12.8 Variant record datatype table

The variant record datatype table shall describe discriminated unions of types; these constructs are also known as variant or choice records. Table 33 illustrates the format that shall be used for the variant record datatype table.

Table 33—Variant record datatype table

Record name	Discriminant			Alternative			Encoding	Semantics
	Name	Type	Enumerator	Name	Type	Semantics		
<variant type>	<name>	<type>	<set 1>	<name 1>	<type 1>	<semantics>	<encoding>	<semantics>
			…	…	…	…		
			<set n>	<name n>	<type n>	<semantics>		
<variant type>	<name>	<type>	<set 1>	<name 1>	<type 1>	<semantics>	<encoding>	<semantics>
			…	…	…	…		
			<set m>	<name m>	<type m>	<semantics>		

The first column (Record name) shall identify the name of the variant record datatype. Record names shall adhere to HLA naming conventions (see 3.3.1).

The second column (Discriminant name) shall identify the name of the discriminant. Discriminant names shall adhere to HLA naming conventions (see 3.3.1).

The third column (Discriminant type) shall identify the datatype of the discriminant. Values in this column shall be names from the enumerated datatype table.

The fourth column (Discriminant enumerator) shall be a set of enumerators that determines the alternative. The enumerators shall be from the enumerated datatype specified in the Discriminant type column. The set shall consist of one or more individual enumerators or enumerator ranges separated by commas, or the special symbol “HLAother.” An enumerator range shall be specified by two enumerators, separated by two periods, and enclosed in square brackets. The enumerator range shall denote all enumerators in the enumerated datatype whose definition in the enumerated datatype table occurs in between the specified enumerators (including the specified enumerators themselves). The symbol “HLAother” shall denote all enumerators in the enumerated datatype that are not explicitly included in any of the other Enumerator entries for the associated record name. The symbol “HLAother” shall only occur at most once for each record name.

The fifth column (Alternative name) shall define the identifier or name for the alternative. Alternative names shall adhere to the HLA naming conventions (see 3.3.1). If no alternative applies to the enumerator described in the Discriminant enumerator column, “NA” shall be entered in this column.

The sixth column (Alternative type) shall identify the datatype of the field. Values in this column shall be names from other datatype tables (simple datatype table, enumerated datatype table, fixed record datatype table, array datatype table, or variant record datatype table). If no alternative applies to the enumerator described in the Discriminant enumerator column, “NA” shall be entered in this column.

The seventh column (Alternative semantics) describe the meaning and use of this alternative. “NA” shall be entered in this column when semantics information is unnecessary.

The eighth column (Encoding) shall describe, in detail, the encoding of the variant record datatype (e.g., the location of the discriminant) as delivered to and received from the RTI. Encoding of fields in a variant record is determined by the datatype specified in the Alternative type column. Encoding of the variant record is determined by the encoding specified in the Encoding column. Record data should be interpreted first through the record encoding scheme and then through the encoding scheme of the field's datatype.

For variant record datatypes that use the predefined variant record encoding, the keyword "HLAvariantRecord" shall be entered in the Encoding column (see 4.12.9). The "HLAvariantRecord" encoding consists of the discriminant followed by the alternative associated with the value of the discriminant. Full details of this predefined encoding are given in 4.12.9.

The ninth column (Semantics) shall describe the meaning and use of this variant record datatype. "NA" shall be entered in this column when semantics information is unnecessary.

An example of the variant record table is provided in Table 34. In this example, the WaiterValue datatype associated with the Efficiency and Cheerfulness attributes of the Waiter class is characterized as a variant record. The discriminant is based on the level of experience accumulated by the employee. For new employees, the alternative *CoursePassed* might be represented as an enumeration indicating whether the employee passed training, until the required tasks are learned. For more experienced employees, the alternative *Rating* might support a more expanded scale (e.g., 1–10) to support management decisions on relative pay increases and possible bonuses. For all other waiter types, i.e., temporary staff (presumably only hired for special functions), the alternative is empty. The discriminant itself is represented as an enumerated datatype, as is illustrated in Table 28.

Table 34—Variant record datatype table example

Record Name	Discriminant			Alternative			Encoding	Semantics
	Name	Type	Enumerator	Name	Type	Semantics		
Waiter Value	Val Index	Experience Level	Trainee	Course Passed	HLAboolean	Ratings scale for employees under training	HLAvariantRecord	Datatype for waiter performance rating value
			[Apprentice .. Senior], Master	Rating	RateScale	Ratings scale for permanent employees		
			HLAother	NA	NA	All others		

4.12.9 Predefined encodings for constructed datatypes

Each of the constructed datatypes (arrays, fixed records, and variant records) has a predefined encoding that describes how the components (i.e., elements of an array, or fields of a fixed or variant record) within the constructed datatype are laid out with respect to each other. These encodings are defined precisely in the appropriate subclauses below. These predefined encoding definitions include a description of how each component in a constructed datatype shall be padded in order to ensure proper byte alignment of the component that follows it. The use of any of the predefined keywords *HLAfixedArray, HLAvariableArray, HLAfixedRecord,* or *HLAvariantRecord* to describe the encoding for a constructed datatype shall indicate adherence to these padding requirements. Object modelers may define and use alternative encoding schemes as needed.

In general, padding bytes shall be added as necessary within a constructed type so that each component is properly aligned. For example, a 32-bit float should be aligned on a 32-bit boundary, and a 64-bit float on a 64-bit boundary. Determining the number of padding bytes needed after a component in a constructed type depends on up to three factors, as follows:

— The offset in bytes of that component from the beginning of the constructed type

— The size in bytes of that component

— The “octet boundary value” of the following component

The octet boundary value for simple datatypes and enumerated datatypes is derived from the type's entry in the basic data representation table. The octet boundary value is defined as the smallest value 2^n, where *n* is a non-negative integer, for which ($8 * 2^n$) is greater than or equal to the size of the datatype in bits. Octet boundary values of the predefined basic data representations are as follows:

Basic representation	Octet boundary value
HLAoctet	1
HLAoctetPairBE	2
HLAinteger16BE	2
HLAinteger32BE	4
HLAinteger64BE	8
HLAfloat32BE	4
HLAfloat64BE	8
HLAoctetPairLE	2
HLAinteger16LE	2
HLAinteger32LE	4
HLAinteger64LE	8
HLAfloat32LE	4
HLAfloat64LE	8

The octet boundary value for a constructed datatype is the maximum octet boundary value of all components within that constructed datatype. For example, a fixed record containing an *HLAboolean* field (based on *HLAinteger32BE),* a field based on *HLAoctet* and a field based on *HLAfloat64BE*, has an octet boundary value of 8. Any constructed datatype that contains this fixed record has an octet boundary value of at least 8, and it may be larger depending on the octet boundary value of its other components.

The following subclauses describe each of the four predefined constructed datatype encodings, including the number of padding bytes to be added after the components of each constructed datatype. If these predefined encodings are used with any datatype whose size in bits is not an integer multiple of 8, padding *bits* shall be added to the datatype to increase its size in bits to the next integer multiple of 8. Padding bits and bytes are always represented by 0s (zeros).

4.12.9.1 HLAfixedRecord

The HLAfixedRecord encoding shall consist of the encoding of each field in sequence, in the order in which they are declared. The first field in a fixed record shall start at offset 0 of the record.

Zero or more padding bytes shall be added to each field, except the last, to ensure that the next field in the record is properly aligned. The number of padding bytes after such a field *i* of a fixed record with this encoding is the smallest nonnegative value of P_i that satisfies the following formula:

$$(\mathrm{Offset}_i + \mathrm{Size}_i + \mathrm{P}_i) \bmod \mathrm{V}_{i+1} = 0$$

where

$Offset_i$ is the offset of the field i of the record (in bytes).

$Size_i$ is the size of the field i of the record (in bytes).

V_{i+1} is the octet boundary value of field (i + 1) of the record.

Padding bytes shall not be added after the last field of the record. The size of a fixed record shall include any added padding bytes.

For example, consider a fixed record using the *HLAfixedRecord* encoding that consists of a field based on *HLAoctet,* an *HLAboolean* field, and a field based on *HLAfloat64BE*, in that order. The encoding calculation is as follows:

Field 1: $(Offset_1 + Size_1 + P_1) \bmod V_2 = 0$
$(0 + 1 + P_1) \bmod 4 = 0$
$(1 + P_1) \bmod 4 = 0$
$P_1 = 3$

Field 2: $(Offset_2 + Size_2 + P_2) \bmod V_3 = 0$
$(4 + 4 + P_2) \bmod 8 = 0$
$(8 + P_2) \bmod 4 = 0$
$P_2 = 0$

Field 3: No padding bytes are added after this, the last, field.

Thus, 3 padding bytes are added after the *HLAoctet*-based field and no padding bytes after the *HLAboolean* field or the *HLAfloat64BE*-based field. The total size of this fixed record is 16 bytes. A graphical representation of this example, showing the contents of each of the 16 bytes with padding bytes represented as 0s, is as follows:

Byte															
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
HLAoctet	0	0	0	HLAboolean				HLAfloat64BE							

Note that if the fixed record consisted of the same three fields, but in the reverse order, no padding bytes are added after any field, and the size of this fixed record is 13 bytes.

4.12.9.2 HLAvariantRecord

The HLAvariantRecord encoding shall consist of the discriminant followed by the alternative associated with the value of the discriminant. The discriminant shall be placed at offset 0 of the record.

If no alternative applies to the value contained in the discriminant, padding bytes shall be added (if required) after the discriminant to ensure that its size in bytes is equal to its octet boundary value.

If there is an alternative for the value contained in the discriminant exists, padding bytes shall be added (if required) after the discriminant to ensure that the alternative is properly aligned. The number of padding bytes after such a discriminant of a variant record with this encoding is the smallest nonnegative value of P that satisfies the following formula:

$(Size + P) \bmod V = 0$

where

Size is the size of the discriminant (in bytes).

V is the maximum of the octet boundary values of the alternatives.

Padding bytes shall not be added after the alternative. The size of the variant record shall include any added padding bytes.

For example, consider a variant record with a discriminant based on the *HLAoctet* representation, one alternative consisting of a field based on the *HLAinteger32BE* representation, and a second alternative consisting of a fixed record containing two fields, both based on the *HLAoctet* representation. The encoding calculation is as follows:

$(Size + P) \bmod V = 0$
$(1 + P) \bmod 4 = 0$
$P = 3$

Thus, 3 bytes of padding are added after the discriminant.

In a case in which the discriminant contains a value that indicates that the *HLAinteger32BE*-based alternative is used, a graphical representation of the variant record is as follows:

Byte							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
HLAoctet	0	0	0	HLAinteger32BE			

In a case in which the discriminant contains a value that indicates that the fixed record alternative is used, a graphical representation of the variant record is as follows:

Byte					
0	**1**	**2**	**3**	**4**	**5**
HLAoctet	0	0	0	HLAoctet	HLAoctet

4.12.9.3 HLAfixedArray

The HLAfixedArray encoding is intended for arrays with fixed cardinality and shall consist of the encoding of each element in sequence. The first element of the array shall be placed starting at offset 0 of the array.

The number of padding bytes after each element *i* of an array, except the last element, with this encoding is the smallest nonnegative value of P_i that satisfies the following formula:

$(Size_i + P_i) \bmod V = 0$

where

$Size_i$ is the size of the ith element of the array (in bytes).

V is the octet boundary value of the element type.

Padding bytes shall not be added after the last element of the array. The size of the fixed array shall include any added padding bytes.

For example, consider an array consisting of a fixed number of elements, each of which is a fixed record encoded using *HLAfixedRecord* and consisting of a field based on the *HLAinteger32BE* representation, followed by a field based on the *HLAoctet* representation.

The size of each element in this array is the same, 5 bytes, as calculated following the rules for the *HLAfixedRecord* encoding. The octet boundary value of each element in this array is 4 bytes, the maximum of the octet boundary values of the two fields of the record. The calculations for the padding after each element, except the last, are as follows:

$(Size_i + P_i) \bmod V = 0$
$(5 + P_i) \bmod 4 = 0$
$P_i = 3$

Thus, 3 bytes of padding are added after each element of the array, except for the last element. A graphical representation of such an array, containing exactly two elements, is as follows:

Byte												
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
HLAinteger32BE				HLA octet	0	0	0	HLAinteger32BE				HLA octet

4.12.9.4 HLAvariableArray

The HLAvariableArray encoding is intended for arrays with variable (including dynamic) cardinality and shall consist of a number of elements component encoded as an *HLAinteger32BE*, followed by the encoding of each element in sequence. The number of elements component shall start at offset 0 of the array.

Padding bytes, if required, shall be added after the number of elements component to ensure that the first element in the array is properly aligned. The number of padding bytes here is the smallest nonnegative value of P that satisfies the formula:

$(4 + P) \bmod V = 0$

where

V is the octet boundary value of the element type.

The number of padding bytes after each element of a variable array is calculated in the same way as for *HLAfixedArray*. The size of the variable array shall include any added padding bytes.

For example, consider an array consisting of a variable number of elements based on the *HLAfloat64BE* representation. The padding calculation for the number of elements component is as follows:

$(4 + P) \bmod V = 0$
$(4 + P) \bmod 8 = 0$
$P = 4$

Thus, 4 bytes of padding are added after the number of elements component. A graphical representation of such an array, containing only one element, is as follows:

Byte															
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
HLAinteger32BE				0	0	0	0	HLAfloat64BE							

As a second example, consider an array, called *VarArray*, consisting of a variable number of elements based on a 24-bit representation. The calculation for padding after the number of elements component is as follows:

$(4 + P) \bmod V = 0$
$(4 + P) \bmod 4 = 0$
$P = 0$

Thus, no padding is added after the number of elements component. The amount of padding after each element, except the last, is calculated as follows (using the formula for *HLAfixedArray* encoding):

$(Size_i + P_i) \bmod V = 0$
$(3 + P_i) \bmod 4 = 0$
$P_i = 1$

Thus, 1 byte of padding is added after each element of the array, except for the last. A graphical representation of such an array, consisting of two elements, is as follows:

Byte										
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
HLAinteger32BE				24-bit value			0	24-bit value		

As a final example, consider a variable length array of *VarArray* elements as described above. In particular, consider an instance that contains two arrays, the first of which contains one element and the second two elements. Here, the entire array of arrays is referred to as the outer array, and each element is referred to as an inner array. Padding after the number of elements component of the outer array is calculated as follows:

$(4 + P) \bmod V = 0$
$(4 + P) \bmod 4 = 0$
$P = 0$

No padding is added after the number of elements component of the outer array. The padding after the first element of the outer array is calculated as follows (using the formula for *HLAfixedArray* encoding):

$(Size_1 + P_1) \bmod V = 0$
$(7 + P_1) \bmod 4 = 0$
$P_1 = 1$

Thus, 1 byte of padding is added after the first element of the outer array, which is the first inner array. No padding is added after the second element of the outer array, which is the second inner array. A graphical representation of this is as follows:

Byte																						
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
HLAinteger32BE				HLAinteger32BE				24-bit value			0	HLAinteger32BE				24-bit value			0	24-bit value		

4.13 Notes table

4.13.1 Purpose/background

Any entry within any of the OMT tables may be annotated with additional descriptive information outside of the immediate table structure. This feature permits users to associate explanatory information with individual table entries to facilitate effective use of the data.

The mechanism for attaching one or more notes to an OMT table entry shall be to include a notes pointer in the appropriate table cell. In the tabular OMT format, this notes pointer shall consist of a uniquely identifying note label (or a series of comma-separated labels) preceded by an asterisk and enclosed by brackets. The notes themselves shall be associated with the note label and included in this table. Note that a single note may be referenced numerous times in OMT tables and that a single OMT table entry may reference numerous notes.

4.13.2 Table format

The template that shall be used for recording notes is illustrated in Table 35.

Table 35—Notes table

Label	Semantics
<label>	<semantics>
<label>	<semantics>
<label>	<semantics>

The first column (Label) shall contain a label for every notes pointer referenced in the object model.

The second column (Semantics) shall contain the explanatory text that constitutes the note.

4.13.3 Inclusion Criteria

FOMs and SOMs describing all federations and federates may include notes wherever such annotation improves the clarity and understandability of the object model.

4.13.4 Example

Table 36 provides an example of the use of the notes feature. Table 8 contains references to these notes.

Table 36—Notes table example

Label	Semantics
1	Merit raises are not provided according to any regular time interval; they are provided on a supervisor's recommendation based on evidence of exceptional effort and performance.
2	Years of service are a factor in any merit raise. This value is only changed on the anniversary of the employee's hire.

5. FOM/SOM lexicon

5.1 Purpose/background

If interoperability among simulations is to be achieved, it is necessary not only to specify the classes of data required by the templates in Clause 4, but also to achieve a common understanding of the semantics of this data. The FOM/SOM lexicon provides a means for federations to define all object classes, interaction classes, object class attributes, and interaction parameters in FOMs, and for individual federates to define these terms in their SOMs.

5.2 Object class definition table

This subclause provides the format for describing HLA object classes. The template that shall be used for this information is provided in Table 37.

Table 37—Object class definition table

Class	Definition
<class>	<definition>
…	…
<class>	<definition>

The first column (Class) shall contain the names of all object classes described in the FOM or SOM. Object class names shall use dot notation as necessary to uniquely identify the object class being defined and may include the full parentage of the class.

The second column (Definition) shall describe the semantics for the object class.

Table 38 provides an example of the use of the table. The object classes are taken from the examples in the object class structure table (see Table 4).

Table 38—Object class definition table example

Class	Definition
Customer	Patron of the restaurant
Employee	A person working for the restaurant
Employee.Dishwasher	Cleaner of plates, pots, and utensils

5.3 Interaction class definition table

This subclause provides the format for describing HLA interactions. The structure that shall be used for this information is provided in Table 39.

Table 39—Interaction class definition table

Class	Definition
<class>	<definition>
…	…
<class>	<definition>

The first column (Class) shall contain the name of each interaction class. Interaction class names shall use dot notation as necessary to uniquely identify the interaction class being defined and may include the full parentage of the class.

The second column (Definition) shall describe the semantics of the interaction class.

Table 40 provides an example of the use of the table. The interaction classes are taken from the examples in the interaction class structure table (see Table 6).

Table 40—Interaction class definition table example

Class	Definition
CustomerTransaction	The base class of interactions between customers and employee
CustomerTransaction.FoodServed	Waiter has served food
CustomerTransaction. FoodServed.DrinkServed	Waiter has served a drink

5.4 Attribute definition table

This subclause provides the format for describing the attributes that characterize object classes. The structure that shall be used for this information is provided in Table 41. A separate row shall be completed for each attribute of an object class.

The first column (Class) shall contain the name of the object class to which a given attribute belongs. Dot notation shall be used as necessary to uniquely identify the object class and may include the full parentage of the class.

Table 41—Attribute definition table

Class	Attribute	Definition
<class>	<attribute>	<definition>
…	…	…
<class>	<attribute>	<definition>

The second column (Attribute) shall contain the name of the attribute.

The third column (Definition) shall describe the characteristic of the object class that this attribute is designed to model.

Table 42 provides an example of the use of the table. The object classes and attributes are taken from the examples in the attribute table (see Table 8).

Table 42—Attribute definition table example

Class	Attribute	Definition
Employee	PayRate	Amount the employee is paid per hour
	YearsOfService	Number of years the employee has worked for the restaurant
	HomeNumber	Employee home phone number
	HomeAddress	Employee home address

5.5 Parameter definition table

This subclause provides the format for describing the parameters that are associated with interaction classes. The structure that shall be used for this information is provided in Table 43. A separate row shall be completed for each parameter of an interaction class.

Table 43—Parameter definition table

Class	Parameter	Definition
<class>	<parameter>	<definition>
…	…	…
<class>	<parameter>	<definition>

The first column (Class) shall contain the name of the interaction class with which a given parameter is associated. Interaction class names shall use dot notation as necessary to uniquely identify the interaction class and may include the full parentage of the class.

The second column (Parameter) shall contain the name of the parameter.

The third column (Definition) shall describe the specific information that this parameter is designed to convey.

Table 44 provides an example of the use of the table. The interaction classes and parameters are taken from the examples in the parameter table (see Table 10).

Table 44—Parameter definition table example

Class	Parameter	Definition
CustomerTransaction.FoodServed. MainCourseServed	TemperatureOk	Whether the food was the correct temperature
	AccuracyOk	Whether the correct food was served
	TimelinessOk	Whether the food was served in a reasonable amount of time

Annex A

(informative)

Table entry notation

The OMT table specifications use a subset of BNF [B1] to specify the types of entries that belong in particular table cells. In BNF, the types of terms to be substituted in the table are enclosed in angle brackets (e.g., *<class>*). Optional entries are enclosed in square brackets (e.g., *[<class> (<p/s>)]*). Any parentheses are terminal characters that should appear as shown. Thus, the basic entry in a cell of the object class structure table, designated by *<class> (<p/s>),* indicates a class name followed by a *Publish/Subscribe* code in parentheses.

Annex B

(normative)

Common normalization functions

The following normalization functions are commonly used and may be referred to directly in dimension tables. The term DUB refers to the dimension upper bound specified for the dimension in the third column of Table 11. Ranges are indicated by two values separated by two periods and surrounded by brackets (indicating a closed range) or parentheses (indicating an open range).

B.1 Linear normalization function

Form linear (domain, dimensionLower, dimensionUpper)

Parameters *domain*: a nonenumerated value known to the federate using the dimension
dimensionLower: the lower bound on the value of *domain*
dimensionUpper: the upper bound on the value of *domain*

Function [(domain - dimensionLower) / (dimensionUpper - dimensionLower)] * (DUB – 1)

B.2 Linear enumerated normalization function

Form linearEnumerated (domain, mappedSet)

Parameters *domain*: an enumerated value known to the federate using the dimension
mappedSet: the enumerated values that *domain* can take (compact on the set of integers)

Function [positionInMappedSet(domain) /(mappedSetLength - 1)] * (DUB – 1)
where
positionInMappedSet(domain) is a function returning the position of *domain* in *mappedSet*, starting at zero
mappedSetLength is the number of elements in *mappedSet*

B.3 Enumerated set normalization function

Form enumeratedSet(domain, mappedSet)

Parameters *domain*: an enumerated value known to the federate using the dimension
mappedSet: the enumerated values that *domain* can take (not necessarily compact on the set of integers)

Function RTIset (domain)
where
RTIset (domain) is a function that maps *mappedSet* onto a range in the interval [0..DUB).

B.4 Logarithmic normalization function

Form logarithmic(domain, domainLower, domainUpper)

Parameters *domain*: a positive, nonenumerated value known to the federate using the dimension.
domainLower: the lower bound on the value of *domain* (must be positive)
domainUpper: the upper bound on the value of *domain* (must be greater than *domainLower*)

Function [log(domain/domainLower) / log(domainUpper/domainLower)] *
(DUB – 1)

B.5 Hyperbolic tangent normalization function

Form: tanh(domain, domainCenter, domainSize)

Parameters: *domain*: a nonenumerated value known to the federate using the dimension
domainCenter: the value of *domain* around which the user desires the greatest precision
domainSize: a generic measure of the distance around the *domainCenter* for which the user desires the relatively high precision

Function: [{tanh([domain - domainCenter]/domainSize) + 1}/2] *
(DUB - 1)

Annex C

(normative)

OMT data interchange format (DIF)

The HLA OMT DIF is a standard data exchange format used to store and transfer HLA FOMs and SOMs between FOM/SOM builders and to initialize the HLA RTI. The DIF is built on a common metamodel that represents the information needed to represent and manage object models created using the HLA OMT standard.

C.1 Notation of the DIF

In order to ensure that the definition of the DIF is unambiguous, it has been formally defined in terms of XML, a formal notation used to describe document specifications. The standard for XML notation is described in REC-XML-19980210.

C.2 HLA OMT DIF DTD declaration

XML calls for using a DTD to specify the valid syntax, structure, and format for documents. The DTD that so describes the OMT DIF is included below.

The XML specification defines two levels of correctness for XML documents: “well-formed” and “valid.” A well-formed XML document is one that follows the syntax constraints of XML; for example, all element opening and closing tags are appropriately nested. A valid XML file is one that is well formed and complies with a DTD. That is, a valid XML document must contain all elements and attributes defined in the DTD and only those elements and attributes. To support the exchange of object models among heterogeneous tools, all OMT DIF-compliant files shall be valid, and thus, well formed.

Because it is often necessary to exchange incomplete OMT documents using the OMT DIF (for federations under development, for example), many completeness and consistency constraints are not included in the OMT DTD. Therefore, much of the individual information is specified as optional in the DTD. However, when complete, an OMT document, in addition to being a valid XML document, shall be consistent and complete with respect to the body of this document.

Several instances exist in which the tabular representation differs from the XML representation, as follows:

— In the tabular representation of object models, object class and interaction class inheritance are represented by columns in which the parent of a class was represented to its left and children to its right. In the XML depiction of object models, inheritance is represented by membership, in which where children of a parent class are depicted as class elements contained within the parent class element.

— In the tabular representation, multiple entries in a field are represented by comma-separated lists. In XML, elements are separated by spaces. This situation exists in notes and dimensions attributes.

— The formatting of notes in XML is different from that presented in the tabular representation in the body of this specification. In the tabular representation, a pointer to a note is indicated by an identifier enclosed in brackets in which the opening bracket is preceded with an asterisk. In XML, every XML attribute has a corresponding note attribute. This attribute shall receive the note pointers for the corresponding value attribute; the text of the note shall be recorded in the notes element.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- HLA.dtd This is version 1.0 of a DTD file to fully define the OMT in XML terms -->
<!ELEMENT objectModel (
              objects?,
              interactions?,
              dimensions?,
              time?,
              tags?,
              synchronizations?,
              transportations?,
              switches?,
              dataTypes?,
              notes?)>
<!ATTLIST objectModel
        DTDversion      CDATA    #FIXED "1516.2"
        name            CDATA    #REQUIRED
        nameNotes       NMTOKENS #IMPLIED
        type            (FOM|SOM) #REQUIRED
        typeNotes       NMTOKENS #IMPLIED
        version         CDATA    #IMPLIED
        versionNotes    NMTOKENS #IMPLIED
        date            CDATA    #IMPLIED
        dateNotes       NMTOKENS #IMPLIED
        purpose         CDATA    #IMPLIED
        purposeNotes    NMTOKENS #IMPLIED
        appDomain       CDATA    #IMPLIED
        appDomainNotes  NMTOKENS #IMPLIED
        sponsor         CDATA    #IMPLIED
        sponsorNotes    NMTOKENS #IMPLIED
        pocName         CDATA    #IMPLIED
        pocNameNotes    NMTOKENS #IMPLIED
        pocOrg          CDATA    #IMPLIED
        pocOrgNotes     NMTOKENS #IMPLIED
        pocPhone        CDATA    #IMPLIED
        pocPhoneNotes   NMTOKENS #IMPLIED
        pocEmail        CDATA    #IMPLIED
        pocEmailNotes   NMTOKENS #IMPLIED
        references      CDATA    #IMPLIED
        referencesNotes NMTOKENS #IMPLIED
        other           CDATA    #IMPLIED
        otherNotes      NMTOKENS #IMPLIED>
  <!ELEMENT objects (objectClass+)>
     <!ELEMENT objectClass (attribute*, objectClass*)>
     <!ATTLIST objectClass
              name           NMTOKEN  #REQUIRED
              nameNotes      NMTOKENS #IMPLIED
              sharing       (Publish|Subscribe|PublishSubscribe|Neither) #IMPLIED
              sharingNotes   NMTOKENS #IMPLIED
              semantics      CDATA    #IMPLIED
              semanticsNotes NMTOKENS #IMPLIED >
        <!ELEMENT attribute EMPTY>
        <!ATTLIST attribute
```

```
                name                 NMTOKEN  #REQUIRED
                nameNotes            NMTOKENS #IMPLIED
                dataType             NMTOKEN  #IMPLIED
                dataTypeNotes        NMTOKENS #IMPLIED
                updateType           (Static|Periodic|Conditional|NA) #IMPLIED
                updateTypeNotes      NMTOKENS #IMPLIED
                updateCondition      CDATA    #IMPLIED
                updateConditionNotes NMTOKENS #IMPLIED
                ownership            (Divest|Acquire|DivestAcquire|NoTransfer)
                                     #IMPLIED
                ownershipNotes       NMTOKENS #IMPLIED
                sharing             (Publish|Subscribe|PublishSubscribe|Neither)
                                     #IMPLIED
                  sharingNotes         NMTOKENS #IMPLIED
                  dimensions           NMTOKENS #IMPLIED
                  dimensionsNotes      NMTOKENS #IMPLIED
                  transportation       NMTOKEN  #IMPLIED
                  transportationNotes  NMTOKENS #IMPLIED
                  order                (Receive|TimeStamp) #IMPLIED
                  orderNotes           NMTOKENS #IMPLIED
                  semantics            CDATA    #IMPLIED
                  semanticsNotes       NMTOKENS #IMPLIED>
<!ELEMENT interactions (interactionClass+)>
   <!ELEMENT interactionClass (parameter*, interactionClass*)>
   <!ATTLIST interactionClass
            name              NMTOKEN  #REQUIRED
            nameNotes         NMTOKENS #IMPLIED
            sharing           (Publish|Subscribe|PublishSubscribe|Neither)
                               #IMPLIED
            sharingNotes      NMTOKENS #IMPLIED
            dimensions        NMTOKENS #IMPLIED
            dimensionsNotes   NMTOKENS #IMPLIED
            transportation    NMTOKEN  #IMPLIED
            transportationNotes NMTOKENS #IMPLIED
            order             (Receive|TimeStamp) #IMPLIED
            orderNotes        NMTOKENS #IMPLIED
            semantics         CDATA    #IMPLIED
            semanticsNotes    NMTOKENS #IMPLIED >
   <!ELEMENT parameter EMPTY>
   <!ATTLIST parameter
            name             NMTOKEN  #REQUIRED
            nameNotes        NMTOKENS #IMPLIED
            dataType         NMTOKEN  #IMPLIED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
<!ELEMENT dimensions (dimension*)>
   <!ELEMENT dimension EMPTY>
   <!ATTLIST dimension
            name               NMTOKEN  #REQUIRED
            nameNotes          NMTOKENS #IMPLIED
            dataType           NMTOKEN  #IMPLIED
            dataTypeNotes      NMTOKENS #IMPLIED
            upperBound         CDATA    #IMPLIED
            upperBoundNotes    NMTOKENS #IMPLIED
            normalization      CDATA    #IMPLIED
            normalizationNotes NMTOKENS #IMPLIED
            value              CDATA    #IMPLIED
            valueNotes         NMTOKENS #IMPLIED>
```

```
<!ELEMENT time (timeStamp?, lookahead?)>
   <!ELEMENT timeStamp EMPTY>
   <!ATTLIST timeStamp
            dataType         NMTOKEN  #IMPLIED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT lookahead EMPTY>
   <!ATTLIST lookahead
            dataType         NMTOKEN  #IMPLIED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
<!ELEMENT tags (updateReflectTag?,
              sendReceiveTag?,
              deleteRemoveTag?,
              divestitureRequestTag?,
              divestitureCompletionTag?,
              acquisitionRequestTag?,
              requestUpdateTag?)>
   <!ELEMENT updateReflectTag EMPTY>
   <!ATTLIST updateReflectTag
            dataType         NMTOKEN  #REQUIRED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT sendReceiveTag EMPTY>
   <!ATTLIST sendReceiveTag
            dataType         NMTOKEN  #REQUIRED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT deleteRemoveTag EMPTY>
   <!ATTLIST deleteRemoveTag
            dataType         NMTOKEN  #REQUIRED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT divestitureRequestTag EMPTY>
   <!ATTLIST divestitureRequestTag
            dataType         NMTOKEN  #REQUIRED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT divestitureCompletionTag EMPTY>
   <!ATTLIST divestitureCompletionTag
            dataType         NMTOKEN  #REQUIRED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT acquisitionRequestTag EMPTY>
   <!ATTLIST acquisitionRequestTag
            dataType         NMTOKEN  #REQUIRED
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT requestUpdateTag EMPTY>
   <!ATTLIST requestUpdateTag
            dataType         NMTOKEN  #REQUIRED
```

```
            dataTypeNotes    NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
<!ELEMENT synchronizations (synchronization+)>
   <!ELEMENT synchronization EMPTY>
   <!ATTLIST synchronization
            label            NMTOKEN  #REQUIRED
            labelNotes       NMTOKENS #IMPLIED
            dataType         NMTOKEN  #IMPLIED
            dataTypeNotes    NMTOKENS #IMPLIED
            capability      (Register|Achieve|RegisterAchieve|NoSynch) #IMPLIED
            capabilityNotes  NMTOKENS #IMPLIED
            semantics        CDATA    #IMPLIED
            semanticsNotes   NMTOKENS #IMPLIED >
<!ELEMENT transportations (transportation+)>
   <!ELEMENT transportation EMPTY>
   <!ATTLIST transportation
            name             NMTOKEN  #REQUIRED
            nameNotes        NMTOKENS #IMPLIED
            description      CDATA    #IMPLIED
            descriptionNotes NMTOKENS #IMPLIED>
   <!ELEMENT switches EMPTY>
   <!ATTLIST switches
            autoProvide                       (Enabled|Disabled) #IMPLIED
            autoProvideNotes                  NMTOKENS #IMPLIED
            conveyRegionDesignatorSets        (Enabled|Disabled) #IMPLIED
            conveyRegionDesignatorSetsNotes   NMTOKENS #IMPLIED
            attributeScopeAdvisory            (Enabled|Disabled) #IMPLIED
            attributeScopeAdvisoryNotes       NMTOKENS #IMPLIED
            attributeRelevanceAdvisory        (Enabled|Disabled) #IMPLIED
            attributeRelevanceAdvisoryNotes   NMTOKENS #IMPLIED
            objectClassRelevanceAdvisory      (Enabled|Disabled) #IMPLIED
            objectClassRelevanceAdvisoryNotes NMTOKENS #IMPLIED
            interactionRelevanceAdvisory      (Enabled|Disabled) #IMPLIED
            interactionRelevanceAdvisoryNotes NMTOKENS #IMPLIED
            serviceReporting                  (Enabled|Disabled) #IMPLIED
            serviceReportingNotes             NMTOKENS #IMPLIED>
<!ELEMENT dataTypes (basicDataRepresentations,
                   simpleDataTypes?,
                   enumeratedDataTypes?,
                   arrayDataTypes?,
                   fixedRecordDataTypes?,
                   variantRecordDataTypes?)>
<!ELEMENT basicDataRepresentations (basicData+)>
   <!ELEMENT basicData EMPTY>
   <!ATTLIST basicData
            name                NMTOKEN  #REQUIRED
            nameNotes           NMTOKENS #IMPLIED
            size                CDATA    #IMPLIED
            sizeNotes           NMTOKENS #IMPLIED
            interpretation      CDATA    #IMPLIED
            interpretationNotes NMTOKENS #IMPLIED
            endian              (Big|Little) #IMPLIED
            endianNotes         NMTOKENS #IMPLIED
            encoding            CDATA    #IMPLIED
            encodingNotes       NMTOKENS #IMPLIED>
<!ELEMENT simpleDataTypes (simpleData+)>
   <!ELEMENT simpleData EMPTY>
   <!ATTLIST simpleData
```

```
         name                NMTOKEN  #REQUIRED
         nameNotes           NMTOKENS #IMPLIED
         representation      NMTOKEN  #IMPLIED
         representationNotes NMTOKENS #IMPLIED
         units               CDATA    #IMPLIED
         unitsNotes          NMTOKENS #IMPLIED
         resolution          CDATA    #IMPLIED
         resolutionNotes     NMTOKENS #IMPLIED
         accuracy            CDATA    #IMPLIED
         accuracyNotes       NMTOKENS #IMPLIED
         semantics           CDATA    #IMPLIED
         semanticsNotes      NMTOKENS #IMPLIED>
<!ELEMENT enumeratedDataTypes (enumeratedData+)>
   <!ELEMENT enumeratedData (enumerator+)>
   <!ATTLIST enumeratedData
         name                NMTOKEN  #REQUIRED
         nameNotes           NMTOKENS #IMPLIED
         representation      NMTOKEN  #IMPLIED
         representationNotes NMTOKENS #IMPLIED
         semantics           CDATA    #IMPLIED
         semanticsNotes      NMTOKENS #IMPLIED >
   <!ELEMENT enumerator EMPTY>
   <!ATTLIST enumerator
         name             NMTOKEN  #REQUIRED
         nameNotes        NMTOKENS #IMPLIED
         values           NMTOKENS  #IMPLIED
         valuesNotes      NMTOKENS #IMPLIED>
<!ELEMENT arrayDataTypes (arrayData+)>
   <!ELEMENT arrayData EMPTY>
   <!ATTLIST arrayData
         name             NMTOKEN  #REQUIRED
         nameNotes        NMTOKENS #IMPLIED
         dataType         NMTOKEN  #IMPLIED
         dataTypeNotes    NMTOKENS #IMPLIED
         cardinality      CDATA    #IMPLIED
         cardinalityNotes NMTOKENS #IMPLIED
         encoding         CDATA    #IMPLIED
         encodingNotes    NMTOKENS #IMPLIED
         semantics        CDATA    #IMPLIED
         semanticsNotes   NMTOKENS #IMPLIED>
<!ELEMENT fixedRecordDataTypes (fixedRecordData+)>
   <!ELEMENT fixedRecordData (field+)>
   <!ATTLIST fixedRecordData
         name             NMTOKEN  #REQUIRED
         nameNotes        NMTOKENS #IMPLIED
         encoding         CDATA    #IMPLIED
         encodingNotes    NMTOKENS #IMPLIED
         semantics        CDATA    #IMPLIED
         semanticsNotes   NMTOKENS #IMPLIED >
   <!ELEMENT field EMPTY>
   <!ATTLIST field
         name             NMTOKEN  #REQUIRED
         nameNotes        NMTOKENS #IMPLIED
         dataType         NMTOKEN  #IMPLIED
         dataTypeNotes    NMTOKENS #IMPLIED
         semantics        CDATA    #IMPLIED
         semanticsNotes   NMTOKENS #IMPLIED>
<!ELEMENT variantRecordDataTypes (variantRecordData+)>
   <!ELEMENT variantRecordData (alternative+)>
```

```
    <!ATTLIST variantRecordData
             name               NMTOKEN  #REQUIRED
             nameNotes          NMTOKENS #IMPLIED
             discriminant       CDATA    #IMPLIED
             discriminantNotes  NMTOKENS #IMPLIED
             dataType           NMTOKEN  #IMPLIED
             dataTypeNotes      NMTOKENS #IMPLIED
             encoding           CDATA    #IMPLIED
             encodingNotes      NMTOKENS #IMPLIED
             semantics          CDATA    #IMPLIED
             semanticsNotes     NMTOKENS #IMPLIED>
    <!ELEMENT alternative EMPTY>
    <!ATTLIST alternative
             enumerator        CDATA    #REQUIRED
             enumeratorNotes   NMTOKENS #IMPLIED
             name              NMTOKEN  #IMPLIED
             nameNotes         NMTOKENS #IMPLIED
             dataType          NMTOKEN  #IMPLIED
             dataTypeNotes     NMTOKENS #IMPLIED
             semantics         CDATA    #IMPLIED
             semanticsNotes    NMTOKENS #IMPLIED>

  <!ELEMENT notes (note+)>
    <!ELEMENT note EMPTY>
    <!ATTLIST note
             name            NMTOKEN  #REQUIRED
             semantics       CDATA    #IMPLIED
             semanticsNotes  NMTOKENS #IMPLIED >
```

C.3 Structure of OMT DIF file

The following presents the structure of an object model document using the OMT DIF. It depicts the form and organization of all of the XML constructs that make up a valid object model document. However, since all of the values are “xxx,” the data itself is not valid.

```
<?xml version="1.0"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel
   DTDversion="1516.2"
   name="xxx"
   nameNotes="xxx"
   type="xxx"
   typeNotes="xxx"
   version="xxx"
   versionNotes="xxx"
   date="xxx"
   dateNotes="xxx"
   purpose="xxx"
   purposeNotes="xxx"
   appDomain="xxx"
   appDomainNotes="xxx"
   sponsor="xxx"
   sponsorNotes="xxx"
   pocName="xxx"
   pocNameNotes="xxx"
   pocOrg="xxx"
   pocOrgNotes="xxx"
```

```
pocPhone="xxx"
pocPhoneNotes="xxx"
pocEmail="xxx"
pocEmailNotes="xxx"
references="xxx"
referencesNotes="xxx"
other="xxx"
otherNotes="xxx">
<objects>
   <objectClass name="HLAobjectRoot">
      <attribute name="HLAprivilegeToDeleteObject"/>
      <objectClass
         name="xxx"
         nameNotes="xxx"
         sharing="xxx"
         sharingNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx">
         <attribute
            name="xxx"
            nameNotes="xxx"
            dataType="xxx"
            dataTypeNotes="xxx"
            updateType="xxx"
            updateTypeNotes="xxx"
            updateCondition="xxx"
            updateConditionNotes="xxx"
            ownership="xxx"
            ownershipNotes="xxx"
            sharing="xxx"
            sharingNotes="xxx"
            dimensions="xxx"
            dimensionsNotes="xxx"
            transportation="xxx"
            transportationNotes="xxx"
            order="xxx"
            orderNotes="xxx"
            semantics="xxx"
            semanticsNotes="xxx"/>
      </objectClass>
   </objectClass>
</objects>
<interactions>
   <interactionClass name="HLAinteractionRoot">
      <interactionClass
         name="xxx"
         nameNotes="xxx"
         sharing="xxx"
         sharingNotes="xxx"
         dimensions="xxx"
         dimensionsNotes="xxx"
         transportation="xxx"
         transportationNotes="xxx"
         order="xxx"
         orderNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx">
         <parameter
            name="xxx"
```

```
            nameNotes="xxx"
            dataType="xxx"
            dataTypeNotes="xxx"
            semantics="xxx"
            semanticsNotes="xxx"/>
      </interactionClass>
    </interactionClass>
  </interactions>
  <dimensions>
    <dimension
      name="xxx"
      nameNotes="xxx"
      dataType="xxx"
      dataTypeNotes="xxx"
      upperBound="xxx"
      upperBoundNotes="xxx"
      normalization="xxx"
      normalizationNotes="xxx"
      value="xxx"
      valueNotes="xxx"/>
  </dimensions>
  <time>
    <timeStamp
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
    <lookahead
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
  </time>
  <tags>
    <updateReflectTag
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
    <sendReceiveTag
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
    <deleteRemoveTag
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
    <divestitureRequestTag
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
    <divestitureCompletionTag
      dataType="xxx"
      dataTypeNotes="xxx"
      semantics="xxx"
      semanticsNotes="xxx"/>
```

```
      <acquisitionRequestTag
         dataType="xxx"
         dataTypeNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx"/>
      <requestUpdateTag
         dataType="xxx"
         dataTypeNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx"/>
   </tags>
   <synchronizations>
      <synchronization
         label="xxx"
         labelNotes="xxx"
         dataType="xxx"
         dataTypeNotes="xxx"
         capability="xxx"
         capabilityNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx"/>
   </synchronizations>
   <transportations>
      <transportation
         name="xxx"
         nameNotes="xxx"
         description="xxx"
         descriptionNotes="xxx"/>
      </transportations>
      <switches
      autoProvide="xxx"
      autoProvideNotes="xxx"
      conveyRegionDesignatorSets="xxx"
      conveyRegionDesignatorSetsNotes="xxx"
      attributeScopeAdvisory="xxx"
      attributeScopeAdvisoryNotes="xxx"
      attributeRelevanceAdvisory="xxx"
      attributeRelevanceAdvisoryNotes="xxx"
      objectClassRelevanceAdvisory="xxx"
      objectClassRelevanceAdvisoryNotes="xxx"
      interactionRelevanceAdvisory="xxx"
      interactionRelevanceAdvisoryNotes="xxx"
      serviceReporting="xxx"
      serviceReportingNotes="xxx"/>
   <dataTypes>
      <basicDataRepresentations>
         <basicData
            name="xxx"
            nameNotes="xxx"
            size="xxx"
            sizeNotes="xxx"
            interpretation="xxx"
            interpretationNotes="xxx"
            endian="xxx"
            endianNotes="xxx"
            encoding="xxx"
            encodingNotes="xxx"/>
      </basicDataRepresentations>
      <simpleDataTypes>
```

```
      <simpleData
         name="xxx"
         nameNotes="xxx"
         representation="xxx"
         representationNotes="xxx"
         units="xxx"
         unitsNotes="xxx"
         resolution="xxx"
         resolutionNotes="xxx"
         accuracy="xxx"
         accuracyNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx"/>
   </simpleDataTypes>
   <enumeratedDataTypes>
      <enumeratedData
         name="xxx"
         nameNotes="xxx"
         representation="xxx"
         representationNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx">
         <enumerator
            name="xxx"
            nameNotes="xxx"
            values="xxx"
            valuesNotes="xxx"/>
      </enumeratedData>
   </enumeratedDataTypes>
   <arrayDataTypes>
      <arrayData
         name="xxx"
         nameNotes="xxx"
         dataType="xxx"
         dataTypeNotes="xxx"
         cardinality="xxx"
         cardinalityNotes="xxx"
         encoding="xxx"
         encodingNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx"/>
   </arrayDataTypes>
   <fixedRecordDataTypes>
      <fixedRecordData
         name="xxx"
         nameNotes="xxx"
         encoding="xxx"
         encodingNotes="xxx"
         semantics="xxx"
         semanticsNotes="xxx">
         <field
            name="xxx"
            nameNotes="xxx"
            dataType="xxx"
            dataTypeNotes="xxx"
            semantics="xxx"
            semanticsNotes="xxx"/>
      </fixedRecordData>
   </fixedRecordDataTypes>
```

```
      <variantRecordDataTypes>
         <variantRecordData
            name="xxx"
            nameNotes="xxx"
            discriminant="xxx"
            discriminantNotes="xxx"
            dataType="xxx"
            dataTypeNotes="xxx"
            encoding="xxx"
            encodingNotes="xxx"
            semantics="xxx"
            semanticsNotes="xxx">
            <alternative
               enumerator="xxx"
               enumeratorNotes="xxx"
               name="xxx"
               nameNotes="xxx"
               dataType="xxx"
               dataTypeNotes="xxx"
               semantics="xxx"
               semanticsNotes="xxx"/>
         </variantRecordData>
      </variantRecordDataTypes>
   </dataTypes>
   <notes>
      <note
         name="xxx"
         semantics="xxx"
         semanticsNotes="xxx"/>
      </notes>
   </objectModel>
```

Annex D

(informative)

OMT DIF SOM example

The HLA OMT DIF is described in Annex C. It describes the format and content of an object model in XML terms. The following is an object model document that encodes the samples describing the Restaurant SOM that are presented in this specification using the DIF. In XML terms, the document is well formed and valid according to the HLA OMT DTD specified in Annex C. MOM elements have not been included in this SOM example since the restaurant federate does not perform federation management functions and, thus, does not interact with MOM elements.

```
<?xml version="1.0"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel
   DTDversion="1516.2"
   name="RestaurantExample"
   type="SOM"
   version="1.0 Alpha"
   date="1998-01-01"
   purpose="Example of an object model for a restaurant federate"
   appDomain="Restaurant operations"
   sponsor="Federated foods"
   pocName="Mr. Joseph Doe"
   pocOrg="Joe's Place"
   pocPhone="1-977-555-1234"
   pocEmail="doej@fedfoods.com"
   references="www.fedfoods.com/restsim.html"
   other="See Mobil International Restaurant Guide for more information">
   <objects>
      <objectClass name="HLAobjectRoot"
         sharing="Neither">
         <attribute name="HLAprivilegeToDeleteObject"
            dataType="NA"
            updateType="NA"
            updateCondition="NA"
            ownership="NoTransfer"
            sharing="Neither"
            dimensions="NA"
            transportation="HLAreliable"
            order="TimeStamp"/>
         <objectClass name="Customer"
            sharing="PublishSubscribe"
            semantics="Patron of the restaurant"/>
         <objectClass name="Bill"
            sharing="PublishSubscribe"
            semantics="Statement of money owed by the customer"/>
         <objectClass name="Order"
            sharing="PublishSubscribe"
            semantics="Specific items requested by a customer"/>
         <objectClass name="Employee"
```

```
sharing="Neither"
semantics="A person working for the restaurant">
<attribute name="PayRate"
   dataType="DollarRate"
   updateType="Conditional"
   updateCondition="Merit increase"
   updateConditionNotes="1 2"
   ownership="DivestAcquire"
   sharing="PublishSubscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="TimeStamp"
   semantics="Amount the employee is paid per hour"/>
<attribute name="YearsOfService"
   dataType="Years"
   updateType="Periodic"
   updateCondition="1/year"
   updateConditionNotes="2"
   ownership="DivestAcquire"
   sharing="PublishSubscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="TimeStamp"
   semantics="Number of years the employee has worked for the
             restaurant"/>
<attribute  name="HomeNumber"
   dataType="HLAASCIIstring"
   updateType="Conditional"
   updateCondition="Employee request"
   ownership="DivestAcquire"
   sharing="PublishSubscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="TimeStamp"
   semantics="Employee home phone number"/>
<attribute  name="HomeAddress"
   dataType="AddressType"
   updateType="Conditional"
   updateCondition="Employee request"
   ownership="DivestAcquire"
   sharing="PublishSubscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="TimeStamp"
   semantics="Employee home address"/>
<objectClass name="Greeter"
   sharing="PublishSubscribe"/>
<objectClass name="Waiter"
   sharing="PublishSubscribe">
<attribute name="Efficiency"
   dataType="WaiterValue"
   updateType="Conditional"
   updateCondition="Performance review"
   ownership="DivestAcquire"
   sharing="PublishSubscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="TimeStamp"
   semantics="Efficiency rating of waiter"/>
```

```
      <attribute name="Cheerfulness"
         dataType="WaiterValue"
         updateType="Conditional"
         updateCondition="Performance review"
         ownership="DivestAcquire"
         sharing="PublishSubscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Cheerfulness of waiter"/>
      <attribute name="State"
         dataType="WaiterTasks"
         updateType="Conditional"
         updateCondition="Work flow"
         ownership="DivestAcquire"
         sharing="PublishSubscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="What the waiter is doing"/>
   </objectClass>
   <objectClass name="Cashier"
      sharing="PublishSubscribe"
      semantics="Employee who collects money"/>
   <objectClass name="Dishwasher"
      sharing="PublishSubscribe"
      semantics="Cleaner of plates, pots, and utensils"/>
   <objectClass name="Cook"
      sharing="PublishSubscribe"
      semantics="Preparer of the meal"/>
   </objectClass>
   <objectClass name="Food"
      sharing="Subscribe"
      semantics="Products served at the restaurant">
      <objectClass name="MainCourse"
         sharing="PublishSubscribe"
         semantics="Principal dish"/>
      <objectClass name="Drink"
         sharing="Subscribe"
         semantics="Liquid accompanyment to meal">
         <attribute  name="NumberCups"
            dataType="DrinkCount"
            updateType="Conditional"
            updateCondition="Customer request"
            ownership="NoTransfer"
            sharing="PublishSubscribe"
            dimensions="BarQuantity"
            transportation="HLAreliable"
            order="TimeStamp"
            semantics="Number of drinks"/>
         <objectClass name="Water"
            sharing="PublishSubscribe"
            semantics="Tap water"/>
         <objectClass name="Coffee"
            sharing="PublishSubscribe"
            semantics="Brewed coffee"/>
         <objectClass name="Soda"
            sharing="PublishSubscribe"
            semantics="Carbonated beverage">
```

```
        <attribute  name="Flavor"
          dataType="FlavorType"
          updateType="Conditional"
          updateCondition="Customer request"
          ownership="NoTransfer"
          sharing="PublishSubscribe"
          dimensions="SodaFlavor BarQuantity"
          transportation="HLAreliable"
          order="TimeStamp"
          semantics="Flavor of soda"/>
      </objectClass>
    </objectClass>
    <objectClass name="Appetizers"
      sharing="Subscribe"
      semantics="Portion served before main course">
      <objectClass name="Soup"
        sharing="Subscribe"
        semantics="Soup course">
        <objectClass name="ClamChowder"
          sharing="PublishSubscribe"
          semantics="Clam chowder">
          <objectClass name="Manhattan"
            sharing="Publish"
            semantics="Manhattan style clam chowder"/>
          <objectClass name="NewEngland"
            sharing="Publish"
            semantics="New England style clam chowder"/>
        </objectClass>
        <objectClass name="BeefBarley"
          sharing="PublishSubscribe"
          semantics="Beef barley soup"/>
      </objectClass>
      <objectClass name="Nachos"
        sharing="PublishSubscribe"
        semantics="Nachos"/>
    </objectClass>
    <objectClass name="Entree"
      sharing="Subscribe"
      semantics="Principal dish">
      <objectClass name="Beef"
        sharing="PublishSubscribe"
        semantics="Beef entree"/>
      <objectClass name="Chicken"
        sharing="PublishSubscribe"
        semantics="Chicken entree"/>
      <objectClass name="Seafood"
        sharing="Subscribe"
        semantics="Seafood entree">
        <objectClass name="Fish"
          sharing="PublishSubscribe"
          semantics="Fish of the day"/>
        <objectClass name="Shrimp"
          sharing="PublishSubscribe"
          semantics="Shrimp scampi"/>
        <objectClass name="Lobster"
          sharing="PublishSubscribe"
          semantics="Lobster thermador"/>
      </objectClass>
      <objectClass name="Pasta"
```

```
                sharing="PublishSubscribe"
                semantics="Pasta entree"/>
          </objectClass>
          <objectClass name="SideDish"
             sharing="Subscribe"
             semantics="ala carte dishes">
             <objectClass name="Corn"
                sharing="PublishSubscribe"
                semantics="Corn side dish"/>
             <objectClass name="Broccoli"
                sharing="PublishSubscribe"
                semantics="Broccoli side dish"/>
             <objectClass name="BakedPotato"
                sharing="PublishSubscribe"
                semantics="Baked potato side dish"/>
          </objectClass>
          <objectClass name="Dessert"
             sharing="Subscribe"
             semantics="Deserts">
             <objectClass name="Cake"
                sharing="PublishSubscribe"
                semantics="Cake dessert"/>
             <objectClass name="IceCream"
                sharing="Subscribe"
                semantics="Ice cream dessert">
                <objectClass name="Chocolate"
                   sharing="PublishSubscribe"
                   semantics="Chocolate ice cream"/>
                <objectClass name="Vanilla"
                   sharing="PublishSubscribe"
                   semantics="vanilla ice cream"/>
             </objectClass>
          </objectClass>
       </objectClass>
    </objectClass>
 </objects>
 <interactions>
    <interactionClass name="HLAinteractionRoot"
       sharing="Neither"
       dimensions="NA"
       transportation="HLAreliable"
       order="TimeStamp">
       <interactionClass name="CustomerTransactions"
          sharing="Publish"
          dimensions="NA"
          transportation="HLAreliable"
          order="TimeStamp"
          semantics="The base class of interactions between customers and
            employee">
          <interactionClass name="CustomerSeated"
             sharing="PublishSubscribe"
             dimensions="NA"
             transportation="HLAreliable"
             order="TimeStamp"
             semantics="Customer has occupied a table"/>
          <interactionClass name="OrderTaken"
             sharing="Publish"
             dimensions="NA"
             transportation="HLAreliable"
```

```
      order="TimeStamp"
      semantics="Waiter has taken customers order">
      <interactionClass name="FromKidsMenu"
         sharing="Publish"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Customer ordered from children menu"/>
      <interactionClass name="FromAdultMenu"
         sharing="Publish"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Customer ordered from normal menu"/>
   </interactionClass>
   <interactionClass name="FoodServed"
      sharing="Publish"
      dimensions="NA"
      transportation="HLAreliable"
      order="TimeStamp"
      semantics="Waiter has served food">
      <interactionClass name="DrinkServed"
         sharing="Publish"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Waiter has served a drink"/>
      <interactionClass name="AppetizerServed"
         sharing="Publish"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Waiter has served appetizers"/>
      <interactionClass name="MainCourseServed"
         sharing="Publish"
         dimensions="WaiterId"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Waiter has served entree">
         <parameter name="TemperatureOk"
            dataType="ServiceStat"
            semantics="Whether the food was the correct temperature"/>
         <parameter  name="AccuracyOk"
            dataType="ServiceStat"
            semantics="Whether the correct food was served"/>
         <parameter  name="TimelinessOk"
            dataType="HLAboolean"
            semantics="Whether the food was served in a reasonable amount
              of time"/>
      </interactionClass>
      <interactionClass name="DessertServed"
         sharing="Publish"
         dimensions="NA"
         transportation="HLAreliable"
         order="TimeStamp"
         semantics="Waiter has served dessert"/>
   </interactionClass>
   <interactionClass name="CustomerPays"
      sharing="Publish"
```

```
            dimensions="NA"
            transportation="HLAreliable"
            order="TimeStamp"
            semantics="Customer has paid bill">
            <interactionClass name="ByCreditCard"
               sharing="Publish"
               dimensions="NA"
               transportation="HLAreliable"
               order="TimeStamp"
               semantics="Credit card payment"/>
            <interactionClass name="ByCash"
               sharing="Publish"
               dimensions="NA"
               transportation="HLAreliable"
               order="TimeStamp"
               semantics="Cash payment"/>
         </interactionClass>
         <interactionClass name="CustomerLeaves"
            sharing="PublishSubscribe"
            dimensions="NA"
            transportation="HLAreliable"
            order="TimeStamp"
            semantics="Customer departs restaurant"/>
      </interactionClass>
   </interactionClass>
</interactions>
<dimensions>
   <dimension name="SodaFlavor"
      dataType="FlavorType"
      upperBound="3"
      normalization="linearEnumerated (Flavor, [Cola, Orange, RootBeer])"
      value="[0 .. 3)"/>
   <dimension name="BarQuantity"
      dataType="DrinkCount"
      upperBound="25"
      normalization="linear (NumberCups, 1, 25)"
      value="[0)"/>
   <dimension name="WaiterId"
      dataType="EmplId"
      upperBound="20"
      normalization="linear (WaiterId, 1, 20)"
      value="Excluded"/>
</dimensions>
<time>
   <timeStamp
      dataType="TimeType"
      semantics="Floating point value expressed in minutes"/>
   <lookahead
      dataType="LAType"
      semantics="Floating point value expressed in minutes (non-negative)"/>
</time>
<tags>
   <updateReflectTag
      dataType="NA"
      semantics="NA"/>
   <sendReceiveTag
      dataType="NA"
      semantics="NA"/>
   <deleteRemoveTag
```

```
      dataType="HLAASCIIstring"
      semantics="Reason for deletion"/>
   <divestitureRequestTag
      dataType="PriorityLevel"
      semantics="High value for immediate transfer"/>
   <divestitureCompletionTag
      dataType="NA"
      semantics="NA"/>
   <acquisitionRequestTag
      dataType="PriorityLevel"
      semantics="High value for immediate transfer"/>
   <requestUpdateTag
      dataType="NA"
      semantics="NA"/>
</tags>
<synchronizations>
   <synchronization
      label="InitialPublish"
      dataType="NA"
      capability="Achieve"
      semantics="Achieved when all classes are published and subscribed, and all
                initially present objects are registered"/>
   <synchronization
      label="InitialUpdate"
      dataType="NA"
      capability="Achieve"
      semantics="Achieved when instance attribute values for all initially
        present objects are updated"/>
   <synchronization
      label="BeginTimeAdvance"
      dataType="NA"
      capability="Achieve"
      semantics="Achieved when time management services are invoked"/>
   <synchronization
      label="PauseExecution"
      dataType="TimeType"
      capability="RegisterAchieve"
      semantics="Achieved when the time advance after the time in the user-
                supplied tag is attained; time advance requests should then
                cease"/>
</synchronizations>
<transportations>
   <transportation
      name="HLAreliable"
      description="Provide reliable delivery of data in the sense that TCP/IP
                  delivers its data reliably"/>
   <transportation
      name="HLAbestEffort"
      description="Make an effort to deliver data in the sense that UDP provides
                  best-effort delivery"/>
   <transportation
      name="LowLatency"
      description="Choose the delivery mechanism that results in the lowest
                  latency from service initiation to callback invocation at
                  the receiving federate"/>
</transportations>
<switches
   autoProvide="Disabled"
   conveyRegionDesignatorSets="Disabled"
```

```
      attributeScopeAdvisory="Enabled"
      attributeRelevanceAdvisory="Enabled"
      objectClassRelevanceAdvisory="Enabled"
      interactionRelevanceAdvisory="Enabled"
      serviceReporting="Disabled"/>
   <dataTypes>
      <basicDataRepresentations>
        <basicData name="HLAinteger16BE"
            size="16"
            interpretation="Integer in the range [-2^15, 2^15 - 1]"
            endian="Big"
            encoding="16-bit two's complement signed integer. The most significant
                      bit contains the sign."/>
        <basicData name="HLAinteger32BE"
            size="32"
            interpretation="Integer in the range [-2^31, 2^31 - 1]"
            endian="Big"
            encoding="32-bit two's complement signed integer. The most
                      significant bit contains the sign."/>
         <basicData name="HLAinteger64BE"
            size="64"
            interpretation="Integer in the range [-2^63, 2^63 - 1]"
            endian="Big"
            encoding="64-bit two's complement signed integer first. The most
                      significant bit contains the sign."/>
         <basicData name="HLAfloat32BE"
            size="32"
            interpretation="Single-precision floating point number"
            endian="Big"
            encoding="32-bit IEEE normalized single-precision format. See IEEE
                      Std 754-1985"/>
         <basicData name="HLAfloat64BE"
            size="64"
            interpretation="Double-precision floating point number"
            endian="Big"
            encoding="64-bit IEEE normalized double-precision format. See IEEE
                      Std 754-1985"/>
         <basicData name="HLAoctetPairBE"
            size="16"
            interpretation="16-bit value"
            endian="Big"
            encoding="Assumed to be portable among hardware devices."/>
        <basicData name="HLAinteger16LE"
            size="16"
            interpretation="Integer in the range [-2^15, 2^15 - 1]"
            endian="Little"encoding="16-bit two's complement signed integer. The
                    most significant bit contains the sign."/>
        <basicData name="HLAinteger32LE"
            size="32"
            interpretation="Integer in the range [-2^31, 2^31 - 1]"
            endian="Little"
            encoding="32-bit two's complement signed integer. The most significant
                      bit contains the sign."/>
         <basicData name="HLAinteger64LE"
            size="64"
            interpretation="Integer in the range [-2^63, 2^63 - 1]"
            endian="Little"
            encoding="64-bit two's complement signed integer first. The most
                      significant bit contains the sign."/>
```

```
      <basicData name="HLAfloat32LE"
         size="32"
         interpretation="Single-precision floating point number"
         endian="Little"
         encoding="32-bit IEEE normalized single-precision format. See IEEE
                  Std 754-1985"/>
      <basicData name="HLAfloat64LE"
         size="64"
         interpretation="Double-precision floating point number"
         endian="Little"
         encoding="64-bit IEEE normalized double-precision format. See IEEE
                  Std 754-1985"/>
      <basicData name="HLAoctetPairLE"
         size="16"
         interpretation="16-bit value"
         endian="Little"
         encoding="Assumed to be portable among hardware devices."/>
      <basicData name="HLAoctet"
         size="8"
         interpretation="8-bit value"
         endian="Big"
         encoding="Assumed to be portable among hardware devices."/>
      <basicData name="UnsignedShort"
         size="16"
         interpretation=" Integer in the range [0, 2^16 - 1]"
         endian="Big"
         encoding="16-bit unsigned integer."/>
   &lt;/basicDataRepresentations&gt;
   &lt;simpleDataTypes&gt;
      <simpleData name="HLAASCIIchar"
         representation="HLAoctet"
         units="NA"
         resolution="NA"
         accuracy="NA"
         semantics="Standard ASCII character (see ANSI Std x3.4-1986)"/>
      <simpleData name="HLAunicodeChar"
         representation="HLAoctetPairBE"
         units="NA"
         resolution="NA"
         accuracy="NA"
         semantics="Unicode UTF-16 character (see The Unicode Standard,
                     Version 3.0)"/>
      <simpleData name="HLAbyte"
         representation="HLAoctet"
         units="NA"
         resolution="NA"
         accuracy="NA"
         semantics="Uninterpreted 8-bit byte"/>
      <simpleData name="TimeType"
         representation="HLAfloat32BE"
         units="Minutes"
         resolution="0.01667"
         accuracy="NA"
         semantics="Time representation"/>
      <simpleData name="LAType"
         representation="HLAfloat32BE"
         units="Minutes"
         resolution="0.01667"
         accuracy="NA"
```

```
            semantics="Time interval (non-negative)"/>
        <simpleData name="DollarRate"
            representation="HLAfloat32BE"
            units="$/hour"
            resolution="0.01"
            accuracy="Perfect"
            semantics="Cost per hour"/>
        <simpleData name="Years"
            representation="HLAinteger32BE"
            units="Years"
            resolution="1"
            accuracy="Perfect"
            semantics="Elapsed time in years"/>
        <simpleData name="DrinkCount"
            representation="UnsignedShort"
            units="Cups"
            resolution="1"
            accuracy="Perfect"
            semantics="Measure of number of drinks"/>
        <simpleData name="EmplId"
            representation="HLAinteger32BE"
            units="NA"
            resolution="1"
            accuracy="Perfect"
            semantics="Employee identifier"/>
        <simpleData name="RateScale"
            representation="HLAinteger32BE"
            units="NA"
            resolution="1"
            accuracy="Perfect"
            semantics="Evaluation of staff where 1 = best"/>
    </simpleDataTypes>
    <enumeratedDataTypes>
        <enumeratedData name="HLAboolean"
            representation="HLAinteger32BE"
            semantics="Standard boolean type">
            <enumerator name="HLAfalse" values="0"/>
            <enumerator name="HLAtrue" values="1"/>
        </enumeratedData>
        <enumeratedData name="PriorityLevel"
            representation="HLAinteger32BE"
            semantics="General three level priority indicator">
            <enumerator name="Low" values="0"/>
            <enumerator name="Medium" values="1"/>
            <enumerator name="High" values="2"/>
        </enumeratedData>
        <enumeratedData name="WaiterTasks"
            representation="HLAinteger32BE"
            semantics="Possible activities of waiters">
            <enumerator name="TakingOrder" values="1"/>
            <enumerator name="Serving" values="2"/>
            <enumerator name="Cleaning" values="3"/>
            <enumerator name="CalculatingBill" values="4"/>
            <enumerator name="Other" values="5"/>
        </enumeratedData>
        <enumeratedData name="FlavorType"
            representation="HLAinteger32BE"
            semantics="Possible flavors of soda">
            <enumerator name="Cola" values="101"/>
```

```
        <enumerator name="Orange" values="102"/>
        <enumerator name="RootBeer" values="103"/>
        <enumerator name="Cream" values="104"/>
      </enumeratedData>
      <enumeratedData name="ExperienceLevel"
        representation="HLAinteger32BE"
        semantics="Level of experience of waiters">
        <enumerator name="Trainee" values="0"/>
        <enumerator name="Apprentice" values="1"/>
        <enumerator name="Journeyman" values="2"/>
        <enumerator name="Senior" values="3"/>
        <enumerator name="Temporary" values="4"/>
        <enumerator name="Master" values="5"/>
      </enumeratedData>
    </enumeratedDataTypes>
    <arrayDataTypes>
      <arrayData name="HLAASCIIstring"
        dataType="HLAASCIIchar"
        cardinality="Dynamic"
        encoding="HLAvariableArray"
        semantics="ASCII string representation"/>
      <arrayData name="HLAunicodeString"
        dataType="HLAunicodeChar"
        cardinality="Dynamic"
        encoding="HLAvariableArray"
        semantics="Unicode string representation"/>
      <arrayData name="HLAopaqueData"
        dataType="HLAbyte"
        cardinality="Dynamic"
        encoding="HLAvariableArray"
        semantics="Uninterpreted sequence of bytes"/>
      <arrayData name="Employees"
        dataType="EmplId"
        cardinality="10"
        encoding="HLAfixedArray"
        semantics="Identifiers of employees currently working"/>
      <arrayData name="AddressBook"
        dataType="AddressType"
        cardinality="Dynamic"
        encoding="An HLAinteger32BE followed by a set of index-value tuples.
                 Each tuple consists of an HLAinteger32BE indicating the
                 array index, followed by the element for that index. The
                 initial HLAinteger32BE indicates the number of index-value
                 pairs to follow, since all array elements need not be
                 included."
        semantics="Collection of all employee addresses"/>
    </arrayDataTypes>
    <fixedRecordDataTypes>
      <fixedRecordData name="ServiceStat"
        encoding="HLAfixedRecord"
        semantics="Check-off on whether the server performed properly on
          elements of the meal">
        <field name="EntreeOk"
          dataType="HLAboolean"
          semantics="Entree status"/>
        <field name="Veggy1Ok"
          dataType="HLAboolean"
          semantics="Vegetable 1 status"/>
        <field name="Veggy2Ok"
```

```
            dataType="HLAboolean"
            semantics="Vegetable 2 status"/>
      </fixedRecordData>
      <fixedRecordData name="AddressType"
         encoding="HLAfixedRecord"
         semantics="Mailing address">
         <field name="Name"
            dataType="HLAASCIIstring"
            semantics="Employee name"/>
         <field name="Street"
            dataType="HLAASCIIstring"
            semantics="Street address"/>
         <field name="City"
            dataType="HLAASCIIstring"
            semantics="City name"/>
         <field name="State"
            dataType="HLAASCIIstring"
            semantics="State abbreviation"/>
         <field name="Zip"
            dataType="HLAASCIIstring"
            semantics="Postal code"/>
      </fixedRecordData>
   </fixedRecordDataTypes>
   <variantRecordDataTypes>
      <variantRecordData name="WaiterValue"
         discriminant="ValIndex"
         dataType="ExperienceLevel"
         encoding="HLAvariantRecord"
         semantics="Datatype for waiter performance rating value">
         <alternative
            enumerator="Trainee"
            name="CoursePassed"
            dataType="HLAboolean"
            semantics="Ratings scale for employees under training"/>
         <alternative
            enumerator="[Apprentice .. Senior], Master"
            name="Rating"
            dataType="RateScale"
            semantics="Ratings scale for permanent employees"/>
         <alternative
            enumerator="HLAother"
            name="NA"
            dataType="NA"
            semantics="All others"/>
      </variantRecordData>
   </variantRecordDataTypes>
</dataTypes>
<notes>
   <note
      name="1"
      semantics="Merit raises are not provided according to any regular time
                 interval; they are provided on a supervisor's recommendation
                 based on evidence of exceptional effort and performance"/>
   <note
      name="2"
      semantics="Years of service are a factor in any merit raise. This value
                 is only changed on the anniversary of the employee's hire."/>
</notes>
</objectModel>
```

Annex E

(informative)

OMT DIF FOM example

NOTE—Copyright to the OMT DIF FOM Code below pertains to the Institute of Electrical and Electronics Engineers, Inc. (IEEE). Copyright © 2001. All rights reserved. Copyright permission to utilize the OMT DIF FOM Code for derivative works may be obtained by contacting the Contracts Administrator, IEEE Standards Activities, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

The HLA OMT DIF is described in Annex C. It describes the format and content of an object model in XML terms. The following is an object model document that encodes a very small FOM using the DIF. In XML terms, the document is well formed and valid according to the HLA OMT DTD specified in Annex C. MOM elements have been included in this example since a FOM is depicted.

```
<?xml version="1.0"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel
   DTDversion="1516.2"
   name="Example"
   type="FOM"
   version="1.0"
   date="2000-04-01"
   purpose="Provide an example of an HLA FOM"
   sponsor="DMSO">
   <objects>
      <objectClass name="HLAobjectRoot"
         sharing="Neither">
         <attribute name="HLAprivilegeToDeleteObject"
            dataType="NA"
            updateType="NA"
            updateCondition="NA"
            ownership="NoTransfer"
            sharing="Neither"
            dimensions="NA"
            transportation="HLAreliable"
            order="TimeStamp"/>
         <objectClass name="UserBaseClass"
            sharing="Neither"
            semantics="This object class is the base of all user-defined object
                       classes">
            <objectClass name="UserSubclass"
               sharing="PublishSubscribe"
               semantics="This is a subclass of UserBaseClass">
               <attribute name="UserAttribute"
                  dataType="UserDatatype"
                  updateType="Static"
                  updateCondition="NA"
                  ownership="NoTransfer"
                  sharing="PublishSubscribe"
                  dimensions="NA"
                  transportation="HLAreliable"
                  order="TimeStamp"
                  semantics="Attribute of UserSubclass"/>
            </objectClass>
```

```
      </objectClass>
      <objectClass name="HLAmanager"
         sharing="Neither"
         semantics="This object class is the root class of all MOM object
                    classes">
         <objectClass name="HLAfederate"
            sharing="Publish"
            semantics="This object class shall contain RTI state variables
                       relating to a joined federate. The RTI shall publish it
                       and shall register one object instance for each joined
                       federate in a federation. Dynamic attributes that shall
                       be contained in an object instance shall be updated
                       periodically, where the period should be determined by
                       an interaction of the class HLAmanager.
                       HLAfederate.HLAadjust.HLAsetTiming. If this value is
                       never set or is set to zero, no periodic up-date shall
                       be performed by the RTI.">
            <attribute name="HLAfederateHandle"
               dataType="HLAhandle"
               updateType="Static"
               updateCondition="NA"
               ownership="NoTransfer"
               sharing="Publish"
               dimensions="Federate"
               transportation="HLAreliable"
               order="Receive"
               semantics="Handle of the joined federate returned by a Join
                          Federation Execution service invocation"/>
            <attribute name="HLAfederateType"
               dataType="HLAunicodeString"
               updateType="Static"
               updateCondition="NA"
               ownership="NoTransfer"
               sharing="Publish"
               dimensions="Federate"
               transportation="HLAreliable"
               order="Receive"
               semantics="Type of the joined federate specified by the joined
                          federate when it joined the federation"/>
            <attribute name="HLAfederateHost"
               dataType="HLAunicodeString"
               updateType="Static"
               updateCondition="NA"
               ownership="NoTransfer"
               sharing="Publish"
               dimensions="Federate"
               transportation="HLAreliable"
               order="Receive"
               semantics="Host name of the computer on which the joined federate
                          is executing"/>
            <attribute name="HLARTIversion"
               dataType="HLAunicodeString"
               updateType="Static"
               updateCondition="NA"
               ownership="NoTransfer"
               sharing="Publish"
               dimensions="Federate"
               transportation="HLAreliable"
               order="Receive"
```

```
   semantics="Version of the RTI software being used"/>
<attribute name="HLAFDDID"
   dataType="HLAunicodeString"
   updateType="Static"
   updateCondition="NA"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Identifier associated with the FDD data used by the
             joined federate"/>
<attribute name="HLAtimeConstrained"
   dataType="HLAboolean"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Whether the time advance of the joined federate is
             constrained by other joined federates"/>
<attribute name="HLAtimeRegulating"
   dataType="HLAboolean"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Whether the joined federate influences the time
             advancement of other joined federates"/>
<attribute name="HLAasynchronousDelivery"
   dataType="HLAboolean"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Whether the RTI shall deliver receive-order messages
             to the joined federate while the joined federate's
             time manager state is not Time Advancing (only matters
             if the joined federate is time-constrained)"/>
<attribute name="HLAfederateState"
   dataType="HLAfederateState"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="State of the joined federate"/>
<attribute name="HLAtimeManagerState"
   dataType="HLAtimeState"
```

```
        updateType="Conditional"
        updateCondition="Service invocation"
        ownership="NoTransfer"
        sharing="Publish"
        dimensions="Federate"
        transportation="HLAreliable"
        order="Receive"
        semantics="State of the joined federate's time manager"/>
    <attribute name="HLAlogicalTime"
        dataType="HLAlogicalTime"
        updateType="Periodic"
        updateCondition="HLAsetTiming.HLAreportPeriod"
        ownership="NoTransfer"
        sharing="Publish"
        dimensions="Federate"
        transportation="HLAreliable"
        order="Receive"
        semantics="Joined federate's logical time. Initial value of this
                  information is initial value of time of the Time
                  Representation Abstract datatype (TRADT)."/>
    <attribute name="HLAlookahead"
        dataType="HLAtimeInterval"
        updateType="Periodic"
        updateCondition="HLAsetTiming.HLAreportPeriod"
        ownership="NoTransfer"
        sharing="Publish"
        dimensions="Federate"
        transportation="HLAreliable"
        order="Receive"
        semantics="Minimum duration into the future that a TSO message
                  will be scheduled. The value shall not be defined if
                  the joined federate is not time-regulating)"/>
    <attribute name="HLAGALT"
        dataType="HLAlogicalTime"
        updateType="Periodic"
        updateCondition="HLAsetTiming.HLAreportPeriod"
        ownership="NoTransfer"
        sharing="Publish"
        dimensions="Federate"
        transportation="HLAreliable"
        order="Receive"
        semantics="Joined federate's Greatest Available Logical Time
                  (GALT). The value shall not be defined if GALT is not
                  defined for the joined federate."/>
    <attribute name="HLALITS"
        dataType="HLAlogicalTime"
        updateType="Periodic"
        updateCondition="HLAsetTiming.HLAreportPeriod"
        ownership="NoTransfer"
        sharing="Publish"
        dimensions="Federate"
        transportation="HLAreliable"
        order="Receive"
        semantics="Joined federate's Least Incoming Time Stamp (LITS).
                  The value shall not be defined if LITS is not defined
                  for the joined federate."/>
    <attribute name="HLAROlength"
        dataType="HLAcount"
        updateType="Periodic"
```

```
   updateCondition="HLAsetTiming.HLAreportPeriod"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Number of RO messages queued for delivery to the
             joined federate."/>
<attribute name="HLATSOlength"
   dataType="HLAcount"
   updateType="Periodic"
   updateCondition="HLAsetTiming.HLAreportPeriod"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Number of TSO messages queued for delivery to the
             joined federate"/>
<attribute name="HLAreflectionsReceived"
   dataType="HLAcount"
   updateType="Periodic"
   updateCondition="HLAsetTiming.HLAreportPeriod"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Total number of reflections received by the joined
             federate."/>
<attribute name="HLAupdatesSent"
   dataType="HLAcount"
   updateType="Periodic"
   updateCondition="HLAsetTiming.HLAreportPeriod"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Total number of updates sent by the joined federate"/>
<attribute name="HLAinteractionsReceived"
   dataType="HLAcount"
   updateType="Periodic"
   updateCondition="HLAsetTiming.HLAreportPeriod"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="Total number of interactions received by the joined
             federate."/>
<attribute name="HLAinteractionsSent"
   dataType="HLAcount"
   updateType="Periodic"
   updateCondition="HLAsetTiming.HLAreportPeriod"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
```

```
      order="Receive"
      semantics="Total number of interactions sent by the joined
                federate. This information shall reflect related DDM
                usage."/>
   <attribute name="HLAobjectsInstancesThatCanBeDeleted"
      dataType="HLAcount"
      updateType="Periodic"
      updateCondition="HLAsetTiming.HLAreportPeriod"
      ownership="NoTransfer"
      sharing="Publish"
      dimensions="Federate"
      transportation="HLAreliable"
      order="Receive"
      semantics="Total number of object instances whose
                HLAprivilegeToDeleteObject attribute is owned by the
                joined federate"/>
   <attribute name="HLAobjectInstancesUpdated"
      dataType="HLAcount"
      updateType="Periodic"
      updateCondition="HLAsetTiming.HLAreportPeriod"
      ownership="NoTransfer"
      sharing="Publish"
      dimensions="Federate"
      transportation="HLAreliable"
      order="Receive"
      semantics="Total number of object instances for which the joined
                federate has invoked the Update Attribute Values
                service."/>
   <attribute name="HLAobjectInstancesReflected"
      dataType="HLAcount"
      updateType="Periodic"
      updateCondition="HLAsetTiming.HLAreportPeriod"
      ownership="NoTransfer"
      sharing="Publish"
      dimensions="Federate"
      transportation="HLAreliable"
      order="Receive"
      semantics="Total number of object instances for which the joined
                federate has had a Reflect Attribute Values service
                invocation."/>
   <attribute name="HLAobjectInstancesDeleted"
      dataType="HLAcount"
      updateType="Periodic"
      updateCondition="HLAsetTiming.HLAreportPeriod"
      ownership="NoTransfer"
      sharing="Publish"
      dimensions="Federate"
      transportation="HLAreliable"
      order="Receive"
      semantics="Total number of times the Delete Object Instance
                service was invoked by the joined federate since the
                federate joined the federation"/>
   <attribute name="HLAobjectInstancesRemoved"
      dataType="HLAcount"
      updateType="Periodic"
      updateCondition="HLAsetTiming.HLAreportPeriod"
      ownership="NoTransfer"
      sharing="Publish"
```

```
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="Total number of times the Remove Object Instance
                    service was invoked for the joined federate since the
                    federate joined the federation."/>
      <attribute name="HLAobjectInstancesRegistered"
         dataType="HLAcount"
         updateType="Periodic"
         updateCondition="HLAsetTiming.HLAreportPeriod"
         ownership="NoTransfer"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="Total number of times the Register Object Instance or
                    Register Object Instance with Region service was
                    invoked by the joined federate since the federate
                    joined the federation."/>
      <attribute name="HLAobjectInstancesDiscovered"
         dataType="HLAcount"
         updateType="Periodic"
         updateCondition="HLAsetTiming.HLAreportPeriod"
         ownership="NoTransfer"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="Total number of times the Discover Object Instance
                    service was invoked for the joined federate since the
                    federate joined the federation"/>
      <attribute name="HLAtimeGrantedTime"
         dataType="HLAmsec"
         updateType="Periodic"
         updateCondition="HLAsetTiming.HLAreportPeriod"
         ownership="NoTransfer"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="Wall clock time duration that the federate has spent
                    in the Time Granted state since the last update of
                    this attribute."/>
      <attribute name="HLAtimeAdvancingTime"
         dataType="HLAmsec"
         updateType="Periodic"
         updateCondition="HLAsetTiming.HLAreportPeriod"
         ownership="NoTransfer"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="Wall clock time duration that the federate has spent
                    in the Time Advancing state since the last update of
                    this attribute."/>
   </objectClass>
   <objectClass name="HLAfederation"
      sharing="Publish"
      semantics="This object class shall contain RTI state variables
```

```
            relating to a federation execution. The RTI shall
            publish it and shall register one object instance for
            the federation execution. It shall not automatically
            update the values of the instance attributes; a joined
            federate shall use a Request Attribute Value Update
            service to obtain values for the instance attributes.">
<attribute name="HLAfederationName"
   dataType="HLAunicodeString"
   updateType="Static"
   updateCondition="NA"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Name of the federation to which the joined federate
            belongs"/>
<attribute name="HLAfederatesinFederation"
   dataType="HLAhandleList"
   updateType="Conditional"
   updateCondition="Federate joins or resigns"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Identifiers of joined federates that are joined to
            the federation"/>
<attribute name="HLARTIversion"
   dataType="HLAunicodeString"
   updateType="Static"
   updateCondition="NA"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Version of the RTI software"/>
<attribute name="HLAFDDID"
   dataType="HLAunicodeString"
   updateType="Static"
   updateCondition="NA"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Identifier associated with the FDD used in the
            relevant Create Federation Execution service
            invocation."/>
<attribute name="HLAlastSaveName"
   dataType="HLAunicodeString"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
```

```
   semantics="Name associated with the last federation state save
             (null if no saves have occurred)"/>
<attribute name="HLAlastSaveTime"
   dataType="HLAlogicalTime"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Logical time at which the last federation state timed
             save occurred. The value shall not be defined if no
             timed saves have occurred."/>
<attribute name="HLAnextSaveName"
   dataType="HLAunicodeString"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Name associated with the next federation state save
             (null if no saves are scheduled)"/>
<attribute name="HLAnextSaveTime"
   dataType="HLAlogicalTime"
   updateType="Conditional"
   updateCondition="Service invocation"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Logical time at which the next federation state timed
             save is scheduled. The value shall not be defined if
             no timed saves are scheduled."/>
<attribute name="HLAautoProvide"
   dataType="HLAswitch"
   updateType="Conditional"
   updateCondition="MOM interaction"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Value of federation-wide Auto Provide Switch. Updated
             when value of switch changes"/>
<attribute name="HLAconveyRegionDesignatorSets"
   dataType="HLAswitch"
   updateType="Conditional"
   updateCondition="MOM interaction"
   ownership="NoTransfer"
   sharing="Publish"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Value of federation-wide Convey Region Designator
             Sets Switch. Updated when value of switch changes"/>
```

```
            </objectClass>
         </objectClass>
      </objectClass>
   </objects>
   <interactions>
      <interactionClass name="HLAinteractionRoot"
         sharing="Neither"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive">
         <interactionClass name="UserInteractionBase"
            sharing="Neither"
            dimensions="NA"
            transportation="HLAreliable"
            order="TimeStamp"
            semantics="Base classs of user-defined interactions">
            <interactionClass name="UserInteractionSubclass"
               sharing="PublishSubscribe"
               dimensions="NA"
               transportation="HLAreliable"
               order="TimeStamp"
               semantics="SubClass of UserInteractionBase">
               <parameter name="UserParameter"
                  dataType="UserDatatype"
                  semantics="Parameter of UserInteractionSubclass"/>
            </interactionClass>
         </interactionClass>
         <interactionClass name="HLAmanager"
            sharing="Neither"
            dimensions="NA"
            transportation="HLAreliable"
            order="Receive"
            semantics="Root class of MOM interactions">
            <interactionClass name="HLAfederate"
               sharing="Neither"
               dimensions="NA"
               transportation="HLAreliable"
               order="Receive"
               semantics="Root class of MOM interactions that deal with a specific
                          joined federate">
               <parameter name="HLAfederate"
                  dataType="HLAhandle"
                  semantics="Handle of the joined federate that was provided when
                             joining."/>
               <interactionClass name="HLAadjust"
                  sharing="Neither"
                  dimensions="NA"
                  transportation="HLAreliable"
                  order="Receive"
                  semantics="Permit a joined federate to adjust the RTI
                             statevariables associated with another joined
                             federate">
               <interactionClass name="HLAsetTiming"
                  sharing="Subscribe"
                  dimensions="NA"
                  transportation="HLAreliable"
                  order="Receive"
                  semantics="Adjust the time period between updates of the
                             HLAmanager.HLAfederate object instance for the
```

```
                specified joined federate. If this interaction is
                never sent, the RTI shall not perform periodic
                updates">
      <parameter name="HLAreportPeriod"
         dataType="HLAseconds"
         semantics="Number of seconds between updates of instance
                   attribute values of the HLAfederate object
                   instance
                   (A zero value causes periodic updates to cease)"/>
   </interactionClass>
   <interactionClass name="HLAmodifyAttributeState"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Modify the ownership state of an attribute of an
                object instance for the specified joined federate.
                If the interaction is used to give ownership of the
                instance attribute to the specified joined federate
                and another joined federate currently owns the
                instance attribute, the owning joined federate
                shall be divested of ownership of the instance
                attribute before ownership is granted to the
                specified joined federate. No notification of
                change of ownership of the instance attribute shall
                be provided to either joined federate. In order for
                ownership of the instance attribute to be granted
                to the specified joined federate, the following
                conditions shall be true:
                - the specified joined federate knows about the object
                  instance
                - the specified joined federate is publishing the
                  corresponding class attribute at the known class
                  of the specified object instance at that joined
                  federate
                - the specified instance attribute is not owned
                  by the RTI (i.e., it's not a predefined attribute
                  of a MOM object class)
                  If one or more of the above conditions are not
                  met, then the interaction shall have no effect and
                  an error shall be reported via an interaction of
                  class HLAmanager. HLAfederate.HLAreport.
                  HLAreportMOMexception.">
         <parameter name="HLAobjectInstance"
            dataType="HLAhandle"
            semantics="Handle of the object instance whose attribute
                      state is being changed"/>
         <parameter name="HLAattribute"
            dataType="HLAhandle"
                     semantics="Handle of the instance attribute
                     whose stateis being changed"/>
         <parameter name="HLAattributeState"
            dataType="HLAownership"
            semantics="New state for the attribute of the object
                      instance"/>
      </interactionClass>
      <interactionClass name="HLAsetServiceReporting"
         sharing="Subscribe"
         dimensions="NA"
```

```
            transportation="HLAreliable"
            order="Receive"
            semantics="Specify whether to report service invocations to0
                      or from the specified joined federate via
                      HLAmanager.HLAfederate.
                      HLAreport.HLAreportServiceInvocation interactions
                      (enable or disable servicereporting). If the
                      specified joined federate is subscribed to the
                      HLAmanager.HLAfederate.HLAreport.
                      HLAreportServiceInvocation interaction, all
                      attempts to enable service reporting for that
                      joined federate by sending an
                      HLAmanager.HLAfederate.HLAadjust.
                      HLAsetServiceReporting interaction with an
                      HLAreportingState parameter value of HLAtrue shall
                      fail and be reported via the normal MOM interaction
                      failure means">
            <parameter name="HLAreportingState"
               dataType="HLAboolean"
               semantics="Whether the RTI should report service
                         invocations (default = HLAfalse)"/>
         </interactionClass>
         <interactionClass name="HLAsetExceptionReporting"
            sharing="Subscribe"
            dimensions="NA"
            transportation="HLAreliable"
            order="Receive"
            semantics="Specify whether the RTI shall report service
                      invocation exceptions via
                      HLAmanager.HLAfederate.HLAreport.
                      HLAreportException interactions">
            <parameter name="HLAreportingState"
               dataType="HLAboolean"
               semantics="Whether the RTI should report exceptions
                         (default = HLAfalse)"/>
         </interactionClass>
      </interactionClass>
      <interactionClass name="HLArequest"
         sharing="Neither"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive"
         semantics="Permit a federate to request RTI data about another
                   federate">
         <interactionClass name="HLArequestPublications"
            sharing="Subscribe"
            dimensions="NA"
            transportation="HLAreliable"
            order="Receive"
            semantics="Request that the RTI send report interactions that
                      contain the publication data of a joined federate.
                      It shall result in one interaction of class
                      HLAmanager. HLAfederate.HLAreport.
                      HLAreportInteractionPublication and one
                      interaction of class HLAmanager.HLAfederate.
                      HLAreport.HLAreportObjectClassPublication for each
                      object class published">
         </interactionClass>
         <interactionClass name="HLArequestSubscriptions"
```

```
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send report interactions that
           contain the subscription data of a joined federate. It
             shall result in one interaction of class HLAmanager.
                HLAfederate.HLAreport.
              HLAreportInteractionSubscription and one
              interaction of class HLAmanager.HLAfederate.
             HLAreport.HLAreportObjectClassSubscription for each
                object class published">
</interactionClass>
<interactionClass name="HLArequestObjectInstancesThatCan
                      BeDeleted"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
              contains the object instances that can be deleted
              at the joined federate. It shall result in one
              interaction of class
              HLAmanager.HLAfederate.HLAreport.
              HLAreportObjectInstancesThatCanBeDeleted">
</interactionClass>
<interactionClass name="HLArequestObjectInstancesUpdated"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
              contains the object instance updating
              responsibility of a joined federate. It shall
              result in one interaction of
                class HLAmanager.HLAfederate.HLAreport.
                HLAreportObjectInstancesUpdated">
</interactionClass>
<interactionClass name="HLArequestObjectInstancesReflected"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
              contains the objects instances for which a joined
              federate has had a Reflect Attribute Values service
              invocation. It shall result in one interaction
              ofclass HLAmanager.HLAfederate.HLAreport.
              HLAreportObjectInstancesReflected">
</interactionClass>
<interactionClass name="HLArequestUpdatesSent"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
              contains the number of updates that a joined
              federate has generated. It shall result in one
              interaction of class
```

```
             HLAmanager.HLAfederate.HLAreport.
             HLAreportUpdatesSent for each transportation type
             that is used to send updates">
</interactionClass>
<interactionClass name="HLArequestInteractionsSent"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
             contains the number of interactions that a joined
             federate has generated. This count shall include
             interactions sent with region. It shall result in
             one interaction of class
             HLAmanager.HLAfederate.HLAreport.
             HLAreportInteractionsSent for each transportation
             type that is used to send interactions">
</interactionClass>
<interactionClass name="HLArequestReflectionsReceived"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
             contains the number of reflections that a joined
             federate has received. It shall result in one
             interaction of class
             HLAmanager.HLAfederate.HLAreport.
             HLAreportReflectionsReceived for each
             transportation type used in receiving
             reflections">
</interactionClass>
<interactionClass name="HLArequestInteractionsReceived"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
             contains the number of interactions that a joined
             federate has received. It shall result in one
             interaction of class
             HLAmanager.HLAfederate.HLAreport.
             HLAreportInteractionsReceived for each
             transportation typeused in receiving
             interactions">
</interactionClass>
<interactionClass name="HLArequestObjectInstanceInformation"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request that the RTI send a report interaction that
             contains the information that a joined federate
             maintains on a single object instance. It shall
             result in one interaction of class
             HLAmanager.HLAfederate.HLAreport.
             HLAreportObjectInstanceInformation">
   <parameter name="HLAobjectInstance"
      dataType="HLAhandle"
```

```
            semantics="Handle of the object instance for which
                      information is being requested"/>
      </interactionClass>
      <interactionClass name="HLArequestSynchronizationPoints"
         sharing="Subscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive"
         semantics="Request that the RTI send a report interaction that
                    contains a list of all in-progress federation
                    synchronization points. It shall result in one
                    interaction class
                    HLAmanager.HLAfederate.HLAreport.
                    HLAreportSynchronizationPoints">
      </interactionClass>
      <interactionClass name="HLArequestSynchronizationPointStatus"
         sharing="Subscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive"
         semantics="Request that the RTI send a report interaction that
                    contains a list that includes each federate (and its
                    synchronization point status) that is associated
                    with a particular synchronization point. It shall
                    result in one interaction of class
                    HLAmanager.HLAfederate.HLAreport.
                    HLAreportSynchronizationPointStaus">
      </interactionClass>
   </interactionClass>
   <interactionClass name="HLAreport"
      sharing="Neither"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Report RTI data about a joined federate. The RTI
                 shall send these interactions in response to
                 interactions of class
                   HLAmanager.HLAfederate.HLArequest.">
      <interactionClass name="HLAreportObjectClassPublication"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="The interaction shall be sent by the RTI in
                    response to an interaction of class
                    HLAmanager.HLAfederate.
                    HLArequest.HLArequestPublications. It shall report
                    the attributes of one object class published by the
                    joined federate. One of these interactions shall be
                    sent for each object class containing attributes
                    that are published by the joined federate">
         <parameter name="HLAnumberOfClasses"
            dataType="HLAcount"
            semantics="The number of object classes for which the joined
                      federate publishes attributes"/>
         <parameter name="HLAobjectClass"
            dataType="HLAhandle"
            semantics="The object class whose publication is being
                      reported"/>
```

```
   <parameter name="HLAattributeList"
      dataType="HLAhandleList"
      semantics="List of handles of HLAobjectClass attributes
                 that the joined federate is publishing"/>
</interactionClass>
<interactionClass name="HLAreportObjectClassSubscription"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in
              response to an interaction of class
              HLAmanager.HLAfederate.
              HLArequest.HLArequestSubscriptions. It shall
              report the attributes of one object class
              subscribed to by the joined federate. One of these
              interactions shall be sent for each object class
              that is subscribed to by the joined federate. This
              information shall reflect related DDM usage.">
   <parameter name="HLAnumberOfClasses"
      dataType="HLAcount"
      semantics="The number of object classes for which the joined
                 federate subscribes to attributes. This
                 information shall reflect related DDM usage."/>
   <parameter name="HLAobjectClass"
      dataType="HLAhandle"
      semantics="The object class whose subscription is being
                 reported"/>
   <parameter name="HLAactive"
      dataType="HLAboolean"
      semantics="Whether the subscription is active"/>
   <parameter name="HLAattributeList"
      dataType="HLAhandleList"
      semantics="List of handles of class attributes to
                 which the joined federate is subscribing."/>
</interactionClass>
<interactionClass name="HLAreportInteractionPublication"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in response
              to an interaction of class HLAmanager.HLAfederate.
              HLArequest.HLArequestPublications. It shall report
              the interaction classes published by the joined
              federate">
   <parameter name="HLAinteractionClassList"
      dataType="HLAhandleList"
      semantics="List of interaction classes that the joined
                 federate is publishing"/>
</interactionClass>
<interactionClass name="HLAreportInteractionSubscription"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in response
              to an interaction of class HLAmanager.HLAfederate.
              HLArequest.HLArequestSubscriptions. It shall
```

```
            report the interaction classes subscribed to by the
            joined federate. This information shall reflect
            related DDM usage.">
   <parameter name="HLAinteractionClassList"
      dataType="HLAinteractionSubList"
      semantics="List of interaction class/subscription type
                 pairs. Each pair consists of the handle of an
                 interaction class that the joined federate is
                 subscribed to and whether the joined federate is
                 actively subscribing. This information shall
                 reflect related DDM usage."/>
</interactionClass>
<interactionClass name="HLAreportObjectInstances
                        ThatCanBeDeleted"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in
            response to an interaction of class
            HLAmanager.HLAfederate.HLArequest.HLArequestObject
            InstancesThatCanBeDeleted. It shall report the
            number of object instances (by registered class of
            the object instances) whose
            HLAprivilegeToDeleteObject attribute areowned by
            the joined federate.">
   <parameter name="HLAobjectInstanceCounts"
      dataType="HLAobjectClassBasedCounts"
      semantics="A list of object instance counts. Each object
                 instance count consists of an object class
                 handle and the number of object instances of
                 that class"/>
</interactionClass>
<interactionClass name="HLAreportObjectInstancesUpdated"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in
            response to an interaction of class
            HLAmanager.HLAfederate.
            HLArequest.HLArequestObjectInstancessUpdated. It
            shall report the number of object instances (by
            registered class of the object instances) for which
            the joined federate has invoked the Update
            Attribute Values service.">
   <parameter name="HLAobjectInstanceCounts"
      dataType="HLAobjectClassBasedCounts"
      semantics="List of object instance counts. Each object
                 instance count consists of an object class
                 handle and the number of object instances of
                 that class."/>
</interactionClass>
<interactionClass name="HLAreportObjectInstancesReflected"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in
```

```
               response to an interaction of class
               HLAmanager.HLAfederate.
                  HLArequest.HLArequestObjectInstancessReflected.
                  It shall report the number of object instances
                  (by registered class of the object instances)
                  for which the joined federate has had a Reflect
                  Attribute Values service invocation.">
      <parameter name="HLAobjectInstanceCounts"
         dataType="HLAobjectClassBasedCounts"
         semantics="List of object instance counts. Each object
                  instance count consists of an object class
                  handle and the number of object instances of
                  that class."/>
   </interactionClass>
   <interactionClass name="HLAreportUpdatesSent"
      sharing="Publish"
      dimensions="Federate"
      transportation="HLAreliable"
      order="Receive"
      semantics="The interaction shall be sent by the RTI in
               response to an interaction of class
               HLAmanager.HLAfederate.
               HLArequest.HLArequestUpdatesSent. It shall report
               the number of updates sent (by registered class of
               the object instances of the updates) by the joined
               federate since the beginning of the federation
               execution. One interaction of this class shall be
               sent by the RTI for each transportation type
               used.">
      <parameter name="HLAtransportation"
         dataType="HLAtransportationName"
         semantics="Transportation type used in sending updates"/>
      <parameter name="HLAupdateCounts"
         dataType="HLAobjectClassBasedCounts"
         semantics="List of update counts. Each update count
                  consists of an object class handle and the
                  number of updates sent of that class"/>
   </interactionClass>
   <interactionClass name="HLAreportReflectionsReceived"
        sharing="Publish"
        dimensions="Federate"
        transportation="HLAreliable"
        order="Receive"
        semantics="The interaction shall be sent by the RTI in
               response to an interaction of class
               HLAmanager.HLAfederate.
               HLArequest.HLArequestReflectionsReceived. It shall
               report the number of reflections received (by
               registered class of the object instances of the
               reflects) by the joined federate since the
               beginning of the federation execution. One
               interaction of this class shall be sent by the RTI
               for each transportation type used.">
      <parameter name="HLAtransportation"
         dataType="HLAtransportationName"
         semantics="Transportation type used in receiving
                  reflections"/>
      <parameter name="HLAreflectCounts"
         dataType="HLAobjectClassBasedCounts"
```

```
      semantics="List of reflection counts. Each reflection
                 count consists of an object class handle and the
                 number of reflections received of that class."/>
</interactionClass>
<interactionClass name="HLAreportInteractionsSent"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in
              response to an interaction of class
              HLAmanager.HLAfederate.
              HLArequest.HLArequestInteractionsSent. It shall
              report the number of interactions sent (by sent
              class of the interactions) by the joined federate
              since the beginning of the federation execution.
              This count shall include interactions sent with
              region. One interaction of this class shall be sent
              by the RTI for each transportation type used.">
   <parameter name="HLAtransportation"
      dataType="HLAtransportationName"
      semantics="Transportation type used in sending
                 interactions"/>
   <parameter name="HLAinteractionCounts"
      dataType="HLAinteractionCounts"
      semantics="List of interaction counts. Each interaction
                 count consists of an interaction class handle
                 and the number of interactions of that class.
                 This information shall reflect related DDM
                 usage."/>
</interactionClass>
<interactionClass name="HLAreportInteractionsReceived"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI in
              response to an interaction of class
              HLAmanager.HLAfederate.
              HLArequest. HLArequestInteractionsReceived. It
              shall report the number of interactions received
              (by sent class of the interactions) by the joined
              federate since the beginning of the federation
              execution. One interaction of this class shall be
              sent by the RTI for each transportation type
              used.">
   <parameter name="HLAtransportation"
      dataType="HLAtransportationName"
      semantics="Transportation type used in receiving
                 interactions"/>
   <parameter name="HLAinteractionCounts"
      dataType="HLAinteractionCounts"
      semantics="List of interaction counts. Each interaction
                 count consists of an interaction class handle
                 and the number of interactions of that class."/>
</interactionClass>
<interactionClass name="HLAreportObjectInstanceInformation"
   sharing="Publish"
   dimensions="Federate"
```

```
      transportation="HLAreliable"
      order="Receive"
      semantics="The interaction shall be sent by the RTI in
                response to an interaction of class
                HLAmanager.HLAfederate.
                HLArequest.HLArequestObjectInstance Information.
                It shall report on a single object instance and
                portray the attributes of that object instance that
                are owned by the joined federate, the registered
                class of the object instance, and the known class
                of the object instance at the joined federate.">
      <parameter name="HLAobjectInstance"
         dataType="HLAhandle"
         semantics="Handle of the object instance for which the
                   interaction was sent"/>
      <parameter name="HLAownedInstanceAttributeList"
         dataType="HLAhandleList"
         semantics="List of the handles of all instance attributes,
                   of the object instance, owned by the joined
                   federate"/>
      <parameter name="HLAregisteredClass"
         dataType="HLAhandle"
         semantics="Handle of the registered class of the object
                   instance"/>
      <parameter name="HLAknownClass"
         dataType="HLAhandle"
         semantics="Handle of the known class of the object
                   instance at the joined federate"/>
   </interactionClass>
   <interactionClass name="HLAreportException"
      sharing="Publish"
      dimensions="Federate"
      transportation="HLAreliable"
      order="Receive"
      semantics="The interaction shall be sent by the RTI when an
                exception occurs as the result of a service
                invocation at the indicated joined federate. This
                interaction shall be sent only if the last
                HLAmanager.HLAfederate.HLAadjust.
                HLAsetExceptionReporting interaction changing the
                HLAreportingState parameter sets the parameter to
                HLAtrue, for the indicated joined federate.">
      <parameter name="HLAservice"
         dataType="HLAunicodeString"
         semantics="Name of the service that raised the exception"/>
      <parameter name="HLAexception"
         dataType="HLAunicodeString"
         semantics="Textual depiction of the exception"/>
   </interactionClass>
   <interactionClass name="HLAreportServiceInvocation"
      sharing="Publish"
      dimensions="Federate ServiceGroup"
      transportation="HLAreliable"
      order="Receive"
      semantics="This interaction shall be sent by the RTI whenever
                an HLA service is invoked, either by the indicated
                joined federate or by the RTI at the indicated
                joined federate, and Service Reporting is Enabled
                for the indicated joined federate. This interaction
```

```
            shall always contain the arguments supplied by the
            service invoker. If the service invocation was
            successful, the interaction also shall contain the
            value returned to the invoker (if the service
            returns a value); otherwise, the interaction also
            shall contain an indication of the exception that
            shall be raised to the invoker.">
   <parameter name="HLAservice"
      dataType="HLAunicodeString"
      semantics="Textual name of the service"/>
   <parameter name="HLAsuccessIndicator"
      dataType="HLAboolean"
      semantics="Whether the service invocation was successful.
                 Exception values are returned along with
                 HLAfalse value"/>
   <parameter name="HLAsuppliedArguments"
      dataType="HLAargumentList"
      semantics="Textual depiction of the arguments supplied in
                 the service invocation"/>
   <parameter name="HLAreturnedArguments"
      dataType="HLAargumentList"
      semantics="Textual depiction of the argument returned by
                 the service invocation (null if the service does
                 not normally return a value or if
                 HLAsuccessIndicator is HLAfalse)"/>
   <parameter name="HLAexception"
      dataType="HLAunicodeString"
      semantics="Textual depiction of the exception raised by
                 this service invocation (null if
                 HLAsuccessIndicator is HLAtrue)"/>
   <parameter name="HLAserialNumber"
      dataType="HLAcount"
      semantics="This is a per-joined federate serial number
                 that shall start at zero and shall increment by
                 1 for each HLAmanager.HLAfederate.HLAreport
                 HLAreportServiceInvocation interaction that
                 represents service invocations to or from the
                 respective joined federate."/>
</interactionClass>
<interactionClass name="HLAreportMOMexception"
   sharing="Publish"
   dimensions="Federate"
   transportation="HLAreliable"
   order="Receive"
   semantics="The interaction shall be sent by the RTI when one
            the following occurs:
            - a MOM interaction without all the necessary
              parameters is sent or
            - an interaction that imitates a federate's
              invocation of an RTI service is sent and not all
              of the service's pre-conditions are met.">
   <parameter name="HLAservice"
      dataType="HLAunicodeString"
      semantics="Name of the service interaction that had a
                 problem or raised the exception"/>
   <parameter name="HLAexception"
      dataType="HLAunicodeString"
      semantics="Textual depiction of the problem or exception"/>
   <parameter name="HLAparameterError"
```

```
            dataType="HLAboolean"
            semantics="HLAtrue if there was an incorrect number of
                    interaction parameters, HLAfalse otherwise"/>
      </interactionClass>
      <interactionClass name="HLAreportSynchronizationPoints"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="The interaction shall be sent by the RTI in
                  response to an interaction of class
                  HLAmanager.HLAfederate.
                  HLArequest.HLArequestSynchronizationPoints. It
                  shall report the list of active synchronization
                  points in the federation execution.">
         <parameter name="HLAsynchPoints"
            dataType="HLAsynchPointList"
            semantics="List of the in progress federation execution
                    synchronization points"/>
      </interactionClass>
      <interactionClass name="HLAreportSynchronizationPointStatus"
         sharing="Publish"
         dimensions="Federate"
         transportation="HLAreliable"
         order="Receive"
         semantics="The interaction shall be sent by the RTI in
                  response to an interaction of class
                  HLAmanager.HLAfederate.
                  HLArequest.HLArequestSynchronizationPointStatus.
                  It shall report the status of a particular
                  synchronization point. This shall be a list that
                  includes each federate (and its synchronization
                  status) that is associated with a particular
                  synchronization point.">
         <parameter name="HLAsynchPointName"
            dataType="HLAunicodeString"
            semantics="Name of a particular synchronization point"/>
         <parameter name="HLAsynchPointFederates"
            dataType="HLAsynchPointFederateList"
            semantics="List of each federate (and its synchronization
                    status) associated with the particular
                    synchronization point"/>
      </interactionClass>
   </interactionClass>
   <interactionClass name="HLAservice"
      sharing="Neither"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="The interaction class shall be acted upon by the RTI.
               These interactions shall invoke HLA services on
               behalf of another joined federate. They shall cause
               the RTI to react as if the service has invoked by that
               other joined federate. If exceptions arise as a result
               of the use of these interactions, they shall be
               reported via the HLAmanager.
               HLAfederate.HLAreport.HLAreportMOMexception
               interaction to all joined federates that subscribe to
               this interaction. There are two ways an error can
```

```
            occur: the sending federate does not provide all the
            required arguments as parameters or the preconditions
            of the spoofed service are not met. Each type of error
            is reported via the HLAMOMreportMOMexception.
            NOTE—These interactions shall have the potential to
            disrupt normal federation execution and should be
            used with great care.">
<interactionClass name="HLAresignFederationExecution"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause the joined federate to resign from the
            federation execution. A joined federate shall be
            able to send this interaction anytime.">
   <parameter name="HLAresignAction"
      dataType="HLAresignAction"
      semantics="Action that the RTI is to take in conjunction
            with the resignation"/>
</interactionClass>
<interactionClass name="HLAsynchronizationPointAchieved"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Mimic the federate's report of achieving a
            synchronization point.">
   <parameter name="HLAlabel"
      dataType="HLAunicodeString"
      semantics="Label associated with the synchronization
            point"/>
</interactionClass>
<interactionClass name="HLAfederateSaveBegun"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Mimic the federate's report of starting a save">
</interactionClass>
<interactionClass name="HLAfederateSaveComplete"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Mimic the joined federate's report of completion
            of a save. A joined federate shall be able to send
            this interaction during a federation save.">
   <parameter name="HLAsuccessIndicator"
      dataType="HLAboolean"
      semantics="Whether the save was successful"/>
</interactionClass>
<interactionClass name="HLAfederateRestoreComplete"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Mimic the joined federate's report of completion
            of a restore.  A joined federate shall be able to
            send this interaction during a federation
```

```
               restore.">
      <parameter name="HLAsuccessIndicator"
         dataType="HLAboolean"
         semantics="Whether the restore was successful"/>
   </interactionClass>
   <interactionClass name="HLApublishObjectClassAttributes"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Set the joined federate's publication status of
               attributes belonging to an object class">
      <parameter name="HLAobjectClass"
         dataType="HLAhandle"
         semantics="Object class for which the joined federate's
                  publication shall change"/>
      <parameter name="HLAattributeList"
         dataType="HLAhandleList"
         semantics="List of handles of attributes of
                  HLAobjectClass, which the federate shall now
                  publish"/>
   </interactionClass>
   <interactionClass name="HLAunpublishObjectClassAttributes"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Cause the joined federate no longer to publish
               attributes of an object class">
      <parameter name="HLAobjectClass"
         dataType="HLAhandle"
         semantics="Object class for which the joined federate's
                  unpublication shall change"/>
      <parameter name="HLAattributeList"
         dataType="HLAhandleList"
         semantics="List of handles of attributes of
                  HLAobjectClass, which the joined federate shall
                  now unpublish"/>
   </interactionClass>
   <interactionClass name="HLApublishInteractionClass"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Set the joined federate's publication status of an
               interaction class">
      <parameter name="HLAinteractionClass"
         dataType="HLAhandle"
         semantics="Interaction class that the joined federate
                  shall publish"/>
   </interactionClass>
   <interactionClass name="HLAunpublishInteractionClass"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Cause the joined federate no longer to publish an
               interaction class">
      <parameter name="HLAinteractionClass"
```

```
            dataType="HLAhandle"
            semantics="Interaction class that the joined federate
                       shall no longer publish"/>
      </interactionClass>
      <interactionClass name="HLAsubscribeObjectClassAttributes"
         sharing="Subscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive"
         semantics="Set the joined federate's subscription status of
                    attributes belonging to an object class">
         <parameter name="HLAobjectClass"
            dataType="HLAhandle"
            semantics="Object class for which the joined federate's
                       subscription shall change"/>
         <parameter name="HLAattributeList"
            dataType="HLAhandleList"
            semantics="List of handles of attributes of HLAobjectClass
                       to which the joined federate shall now
                       subscribe"/>
         <parameter name="HLAactive"
            dataType="HLAboolean"
            semantics="Whether the subscription is active"/>
      </interactionClass>
      <interactionClass name="HLAunsubscribeObjectClassAttributes"
         sharing="Subscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive"
         semantics="Cause the joined federate no longer to subscribe to
                    attributes of an object class">
         <parameter name="HLAobjectClass"
            dataType="HLAhandle"
            semantics="Object class for which the joined federate's
                       subscription shall change"/>
         <parameter name="HLAattributeList"
            dataType="HLAhandleList"
            semantics="List of handles of attributes of HLAobjectClass
                       to which the joined federate shall now
                       unsubscribe"/>
      </interactionClass>
      <interactionClass name="HLAsubscribeInteractionClass"
         sharing="Subscribe"
         dimensions="NA"
         transportation="HLAreliable"
         order="Receive"
         semantics="Set the joined federate's subscription status to an
                    interaction class.">
         <parameter name="HLAinteractionClass"
            dataType="HLAhandle"
            semantics="Interaction class to which the federate shall
                       subscribe"/>
         <parameter name="HLAactive"
            dataType="HLAboolean"
            semantics="Whether the subscription is active"/>
      </interactionClass>
      <interactionClass name="HLAunsubscribeInteractionClass"
         sharing="Subscribe"
         dimensions="NA"
```

```
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause the joined federate no longer to subscribe
             to an interaction class">
   <parameter name="HLAinteractionClass"
      dataType="HLAhandle"
      semantics="Interaction class to which the joined federate
                will no longer be subscribed"/>
</interactionClass>
<interactionClass name="HLAdeleteObjectInstance"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause an object instance to be deleted from the
             federation.">
   <parameter name="HLAobjectInstance"
      dataType="HLAhandle"
      semantics="Handle of the object instance that is to be
                deleted"/>
   <parameter name="HLAtag"
      dataType="HLAopaqueData"
      semantics="Tag associated with the deletion"/>
   <parameter name="HLAtimeStamp"
      dataType="HLAlogicalTime"
      semantics="Time stamp of the deletion (optional)"/>
</interactionClass>
<interactionClass name="HLAlocalDeleteObjectInstance"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Inform the RTI that it shall treat the specified
             object instance as if thejoined federate did not
             know about the object instance.">
   <parameter name="HLAobjectInstance"
        dataType="HLAhandle"
        semantics="Handle of the object instance that is to be
                deleted"/>
</interactionClass>
<interactionClass name="HLAchangeAttributeTransportationType"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Change the transportation type used by the joined
             federate when sending attributes belonging to the
             object instance">
   <parameter name="HLAobjectInstance"
      dataType="HLAhandle"
      semantics="Handle of the object instance whose attribute
                transportation type is to be changed"/>
   <parameter name="HLAattributeList"
      dataType="HLAhandleList"
      semantics="List of the handles of instance attributes
                whose transportation type is to be changed"/>
   <parameter name="HLAtransportation"
      dataType="HLAtransportationName"
      semantics="Transportation type to be used for updating
```

```
                  instance attributes in the list"/>
</interactionClass>
<interactionClass name="HLAchangeInteractionTransportationType"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Change the transportation type used by the joined
             federate when sending a class of interaction">
   <parameter name="HLAinteractionClass"
      dataType="HLAhandle"
      semantics="Interaction class whose transportation type is
                changed by this service invocation"/>
   <parameter name="HLAtransportation"
      dataType="HLAtransportationName"
      semantics="Transportation type to be used for sending the
                interaction class"/>
</interactionClass>
<interactionClass name="HLAunconditionalAttribute
                       OwnershipDivestiture"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause the ownership of attributes of an object
             instance to be unconditionally divested by the
             joined federate">
   <parameter name="HLAobjectInstance"
      dataType="HLAhandle"
      semantics="Handle of the object instance whose attributes'
                ownership is to be divested"/>
   <parameter name="HLAattributeList"
      dataType="HLAhandleList"
      semantics="List of handles of instance attributes
                belonging to HLAobjectInstance whose ownership
                is to be divested by the joined federate"/>
</interactionClass>
<interactionClass name="HLAenableTimeRegulation"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause the joined federate to begin regulating the
             logical time of other joined federates">
   <parameter name="HLAlookahead"
      dataType="HLAtimeInterval"
      semantics="Lookahead to be used by the joined federate
                while regulating other joined federates"/>
</interactionClass>
<interactionClass name="HLAdisableTimeRegulation"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause the joined federate to cease regulating the
             logical time of other joined federates">
</interactionClass>
<interactionClass name="HLAenableTimeConstrained"
   sharing="Subscribe"
```

```
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Cause the logical time of the joined federate to
                 begin being constrained by the logical times of
                 other joined federates">
   </interactionClass>
   <interactionClass name="HLAdisableTimeConstrained"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Cause the logical time of the joined federate to
                 cease being constrained by the logical times of
                 other joined federates">
   </interactionClass>
   <interactionClass name="HLAtimeAdvanceRequest"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Request an advance of the joined federate's
                 logical time on behalf of the joined federate, and
                 release zero or more messages for delivery to the
                 joined federate">
      <parameter name="HLAtimeStamp"
         dataType="HLAlogicalTime"
         semantics="Time stamp requested"/>
   </interactionClass>
   <interactionClass name="HLAtimeAdvanceRequestAvailable"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Request an advance of the joined federate's
                 logical time, on behalf of the joined federate, and
                 release zero or more messages for delivery to the
                 joined federate">
      <parameter name="HLAtimeStamp"
         dataType="HLAlogicalTime"
         semantics="Time stamp requested"/>
   </interactionClass>
   <interactionClass name="HLAnextMessageRequest"
      sharing="Subscribe"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Request the logical time of the joined federate to
                 be advanced to the time stamp of the next TSO
                 message that shall be delivered to the joined
                 federate, provided that the message shall have a
                 time stamp no greater than the logical time
                 specified in the request, and release zero
                 or more messages for delivery to the joined
                 federate.">
      <parameter name="HLAtimeStamp"
         dataType="HLAlogicalTime"
         semantics="Time stamp requested"/>
   </interactionClass>
```

```
<interactionClass name="HLAnextMessageRequestAvailable"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request the logical time of the joined federate to
             be advanced to the time stamp of the next TSO
             message that shall be delivered to the joined
             federate, provided that the message shall have a
             time stamp no greater than the logical time
             specified in the request, and release zero or more
             messages for delivery to the joined federate.">
   <parameter name="HLAtimeStamp"
      dataType="HLAlogicalTime"
      semantics="Time stamp requested"/>
</interactionClass>
<interactionClass name="HLAflushQueueRequest"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Request the logical time of the joined federate to
             be advanced as far as possible, provided that the
             time stamp is less than or equal to the logical
             time specified in the request. All TSO and RO
             messages shall be delivered to the joined
             federate.">
   <parameter name="HLAtimeStamp"
      dataType="HLAlogicalTime"
      semantics="Time stamp requested"/>
</interactionClass>
<interactionClass name="HLAenableAsynchronousDelivery"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Cause the RTI to deliver receive-order messages to
             the joined federate at any time, even if the joined
             federate is time-constrained.">
</interactionClass>
<interactionClass name="HLAdisableAsyncrhonousDelivery"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="When the joined federate is time-constrained,
             cause the RTI to deliver receive-order messages to
             the joined federate only when its time manager
             state is Time Advancing.">
</interactionClass>
<interactionClass name="HLAmodifyLookahead"
   sharing="Subscribe"
   dimensions="NA"
   transportation="HLAreliable"
   order="Receive"
   semantics="Change the lookahead value used by the joined
             federate">
   <parameter name="HLAlookahead"
      dataType="HLAtimeInterval"
```

```
            semantics="New value for lookahead"/>
        </interactionClass>
        <interactionClass name="HLAchangeAttributeOrderType"
          sharing="Subscribe"
          dimensions="NA"
          transportation="HLAreliable"
          order="Receive"
          semantics="Change the ordering type used by the joined
                     federate when sending attributes belonging to the
                     object instance">
          <parameter name="HLAobjectInstance"
            dataType="HLAhandle"
            semantics="Handle of the object instance whose attribute
                       order type is to be changed"/>
          <parameter name="HLAattributeList"
            dataType="HLAhandleList"
            semantics="List of the handles of instance attributes
                       whose order type is to be changed"/>
          <parameter name="HLAsendOrder"
            dataType="HLAorderType"
            semantics="Order type to be used for sending the instance
                       attribute list"/>
        </interactionClass>
        <interactionClass name="HLAchangeInteractionOrderType"
            sharing="Subscribe"
            dimensions="NA"
            transportation="HLAreliable"
            order="Receive"
            semantics="Change the order type used by the joined
                       federate when sending a class of interaction">
          <parameter name="HLAinteractionClass"
            dataType="HLAhandle"
            semantics="Interaction class whose order type is changed by
                       this service invocation"/>
          <parameter name="HLAsendOrder"
            dataType="HLAorderType"
            semantics="Order type to be used for sending the
                       interaction class"/>
        </interactionClass>
      </interactionClass>
    </interactionClass>
    <interactionClass name="HLAfederation"
      sharing="Neither"
      dimensions="NA"
      transportation="HLAreliable"
      order="Receive"
      semantics="Root class of MOM interactions that deal with a specific
                 federation execution.">
      <interactionClass name="HLAadjust"
        sharing="Neither"
        dimensions="NA"
        transportation="HLAreliable"
        order="Receive"
        semantics="Permit a federate to adjust the RTI state variables
                   associated with a federation execution.">
        <interactionClass name="HLAsetSwitches"
          sharing="Subscribe"
          dimensions="NA"
          transportation="HLAreliable"
```

```
                    order="Receive"
                    semantics="Set the values of several HLA switches.">
                    <parameter name="HLAautoProvide"
                       dataType="HLAswitch"
                       semantics="Set the federation-wide Auto Provide Switch to
                                  the provided value (this parameter is required
                                  only when a change of this switch value is
                                  needed)."/>
                    <parameter name="HLAconveyRegionDesignatorSets"
                       dataType="HLAswitch"
                       semantics="Set the federation-wide Convey Region
                                  Designator Sets Switch to the provided value
                                  (this parameter is required only when a change
                                  of this switch value is needed)."/>
                 </interactionClass>
              </interactionClass>
           </interactionClass>
        </interactionClass>
     </interactionClass>
  </interactions>
  <dimensions>
     <dimension name="Federate"
        dataType="HLAfederateHandle"
        upperBoundNotes="MOM1"
        normalization="Normalize Federate Handle service"
        value="Excluded"/>
     <dimension name="ServiceGroup"
        dataType="HLAserviceGroupName"
        upperBound="7"
        normalization="Normalize Service Group service"
        value="Excluded"/>
  </dimensions>
  <time>
     <timeStamp
        dataType="NA"/>
     <lookahead
        dataType="NA"/>
  </time>
  <transportations>
     <transportation
        name="HLAreliable"
        description="Provide reliable delivery of data in the sense that TCP/IP
                     delivers its data reliably"/>
     <transportation
        name="HLAbestEffort"
        description="Make an effort to deliver data in the sense that UDP
                     provides best-effort delivery"/>
  </transportations>
  <switches
     autoProvide="Enabled"
     conveyRegionDesignatorSets="Enabled"
     attributeScopeAdvisory="Enabled"
     attributeRelevanceAdvisory="Enabled"
     objectClassRelevanceAdvisory="Enabled"
     interactionRelevanceAdvisory="Enabled"
     serviceReporting="Disabled"/>
  <dataTypes>
     <basicDataRepresentations>
        <basicData name="HLAinteger16BE"
```

```
    size="16"
    interpretation="Integer in the range [-2^15, 2^15 - 1]"
    endian="Big"
    encoding="16-bit two's complement signed integer. The most
              significant bit contains the sign."/>
<basicData name="HLAinteger32BE"
    size="32"
    interpretation="Integer in the range [-2^31, 2^31 - 1]"
    endian="Big"
    encoding="32-bit two's complement signed integer. The most
              significant bit contains the sign."/>
<basicData name="HLAinteger64BE"
   size="64"
   interpretation="Integer in the range [-2^63, 2^63 - 1]"
   endian="Big"
   encoding="64-bit two's complement signed integer first. The most
             significant bit contains the sign."/>
<basicData name="HLAfloat32BE"
   size="32"
   interpretation="Single-precision floating point number"
   endian="Big"
   encoding="32-bit IEEE normalized single-precision format. See IEEE
             Std 754-1985"/>
<basicData name="HLAfloat64BE"
   size="64"
   interpretation="Double-precision floating point number"
   endian="Big"
   encoding="64-bit IEEE normalized double-precision format. See IEEE
             Std 754-1985"/>
<basicData name="HLAoctetPairBE"
   size="16"
   interpretation="16-bit value"
   endian="Big"
   encoding="Assumed to be portable among hardware devices."/>
<basicData name="HLAinteger16LE"
    size="16"
    interpretation="Integer in the range [-2^15, 2^15 - 1]"
    endian="Little"
    encoding="16-bit two's complement signed integer. The most
              significant bit contains the sign."/>
<basicData name="HLAinteger32LE"
    size="32"
    interpretation="Integer in the range [-2^31, 2^31 - 1]"
    endian="Little"
    encoding="32-bit two's complement signed integer. The most
              significant bit contains the sign."/>
<basicData name="HLAinteger64LE"
   size="64"
   interpretation="Integer in the range [-2^63, 2^63 - 1]"
   endian="Little"
   encoding="64-bit two's complement signed integer first. The most
             significant bit contains the sign."/>
<basicData name="HLAfloat32LE"
   size="32"
   interpretation="Single-precision floating point number"
   endian="Little"
   encoding="32-bit IEEE normalized single-precision format. See IEEE
             Std 754-1985"/>
<basicData name="HLAfloat64LE"
```

```
        size="64"
        interpretation="Double-precision floating point number"
        endian="Little"
        encoding="64-bit IEEE normalized double-precision format. See IEEE
                  Std 754-1985"/>
     <basicData name="HLAoctetPairLE"
        size="16"
        interpretation="16-bit value"
        endian="Little"
        encoding="Assumed to be portable among hardware devices."/>
     <basicData name="HLAOctet"
         size="8"
         interpretation="8-bit value"
         endian="Big"
         encoding="Assumed to be portable among hardware devices."/>
   </basicDataRepresentations>
   <simpleDataTypes>
      <simpleData name="HLAASCIIchar"
         representation="HLAoctet"
         semantics="Standard ASCII character (see ANSI Std X3.4-1986)"/>
      <simpleData name="HLAunicodeChar"
         representation="HLAoctetPairBE"
         units="NA"
         resolution="NA"
         accuracy="NA"
         semantics="Unicode UTF-16 character (see The Unicode Standard,
                    Version 3.0)"/>
      <simpleData name="HLAbyte"
         representation="HLAoctet"
         semantics="Uninterpreted 8-bit byte"/>
      <simpleData name="HLAcount"
         representation="HLAinteger32BE"/>
      <simpleData name="HLAseconds"
         representation="HLAinteger32BE"
         units="seconds"/>
     <simpleData name="HLAmsec"
         representation="HLAinteger32BE"
         units="milliseconds"/>
      <simpleData name="HLAfederateHandle"
         representation="HLAinteger32BE"
         semantics="The type of the argument to Normalize Federate Handle
                    service. This is a pointer to an RTI-defined programming
                    language object, not an integer 32"/>
      <simpleData name="UserDataType"
         representation="HLAinteger32BE"
         units="furlongs"/>
  </simpleDataTypes>
  <enumeratedDataTypes>
     <enumeratedData name="HLAboolean"
        representation="HLAinteger32BE"
        semantics="Standard boolean type">
        <enumerator name="HLAfalse" values="0"/>
        <enumerator name="HLAtrue" values="1"/>
     </enumeratedData>
     <enumeratedData name="HLAfederateState"
        representation="HLAinteger32BE"
        semantics="State of the federate">
        <enumerator name="ActiveFederate" values="1"/>
        <enumerator name="FederateSaveInProgress" values="3"/>
```

```
      <enumerator name="FederateRestoreInProgress" values="5"/>
    </enumeratedData>
    <enumeratedData name="HLAtimeState"
      representation="HLAinteger32BE"
      semantics="State of time advancement">
      <enumerator name="TimeGranted" values="0"/>
      <enumerator name="TimeAdvancing" values="1"/>
    </enumeratedData>
    <enumeratedData name="HLAownership"
      representation="HLAinteger32BE">
      <enumerator name="Unowned" values="0"/>
      <enumerator name="Owned" values="1"/>
    </enumeratedData>
    <enumeratedData name="HLAresignAction"
      representation="HLAinteger32BE"
      semantics="Action to be performed by RTI in conjunction with
                resignation">
      <enumerator name="DivestOwnership" values="1"/>
      <enumerator name="DeleteObjectInstances" values="2"/>
      <enumerator name="CancelPendingAcquisitions" values="3"/>
      <enumerator name="DeleteObjectInstancesThenDivestOwnership"
                      values="4"/>
      <enumerator name="CancelPendingAcquisitionsThenDeleteObject
                      InstancesThenDivestOwnership" values="5"/>
      <enumerator name="NoAction" values="6"/>
    </enumeratedData>
    <enumeratedData name="HLAorderType"
      representation="HLAinteger32BE"
      semantics="Order type to be used for sending attributes or
                interactions">
      <enumerator name="Receive" values="0"/>
      <enumerator name="TimeStamp" values="1"/>
    </enumeratedData >
    <enumeratedData name="HLAswitch"
      representation="HLAinteger32BE">
      <enumerator name="Enabled" values="1"/>
      <enumerator name="Disabled" values="0"/>
    </enumeratedData>
    <enumeratedData name="HLAsynchPointStatus"
      representation="HLAinteger32BE"
      semantics="Joined federate synchronization point status">
      <enumerator name="NoActivity" values="0"/>
      <enumerator name="AttemptingToRegisterSynchPoint" values="1"/>
      <enumerator name="MovingToSynchPoint" values="2"/>
      <enumerator name="WaitingForRestOfFederation" values="3"/>
    </enumeratedData>
    <enumeratedData name="HLAserviceGroupName"
      representation="HLAinteger32BE"
      semantics="Service group identifier">
      <enumerator name="FederationManagement" values="0"/>
      <enumerator name="DeclarationManagement" values="1"/>
      <enumerator name="ObjectManagement" values="2"/>
      <enumerator name="OwnershipManagement" values="3"/>
      <enumerator name="TimeManagement" values="4"/>
      <enumerator name="DataDistributionManagement" values="5"/>
      <enumerator name="SupportServices" values="6"/>
    </enumeratedData>
  </enumeratedDataTypes>
  <arrayDataTypes>
```

```
<arrayData name="HLAASCIIstring"
   dataType="HLAASCIIchar"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="ASCII String representation"/>
<arrayData name="HLAunicodeString"
   dataType="HLAunicodeChar"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="Unicode string representation"/>
<arrayData name="HLAopaqueData"
   dataType="HLAbyte"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="Uninterpreted sequence of bytes"/>
<arrayData name="HLAhandle"
   dataType="HLAbyte"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="Encoded value of a handle. The encoding is based on the
              type of handle"/>
<arrayData name="HLAtransportationName"
   dataType="HLAunicodeChar"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="String whose legal value shall be a name from any row in
              the OMT transportation table (IEEE Std 1516.2-2000"/>
<arrayData name="HLAlogicalTime"
   dataType="HLAbyte"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="An encoded logical time. An empty array shall indicate
              that the values is not defined"/>
<arrayData name="HLAtimeInterval"
   dataType="HLAbyte"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="An encoded logical time interval. An empty array shall
              indicate that the values is not defined"/>
<arrayData name="HLAhandleList"
   dataType="HLAhandle"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="List of encoded handles"/>
<arrayData name="HLAinteractionSubList"
   dataType="HLAinteractionSubscription"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="List of interaction subscription indicators"/>
<arrayData name="HLAargumentList"
   dataType="HLAunicodeString"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="List of arguments"/>
<arrayData name="HLAobjectClassBasedCounts"
   dataType="HLAobjectClassBasedCount"
   cardinality="Dynamic"
   encoding="HLAvariableArray"
   semantics="Counts of various items based on object class"/>
```

```
        <arrayData name="HLAinteractionCounts"
           dataType="HLAinteractionCount"
           cardinality="Dynamic"
           encoding="HLAvariableArray"
           semantics="List of interaction counts"/>
        <arrayData name="HLAsynchPointList"
           dataType="HLAunicodeString"
           cardinality="Dynamic"
           encoding="HLAvariableArray"
           semantics="List of names of synchronization points"/>
        <arrayData name="HLAsynchPointFederateList"
           dataType="HLAsynchPointFederate"
           cardinality="Dynamic"
           encoding="HLAvariableArray"
           semantics="List of joined federates and the synchronization status of
                     each"/>
     </arrayDataTypes>
     <fixedRecordDataTypes>
        <fixedRecordData name="HLAinteractionSubscription"
           encoding="HLAfixedRecord"
           semantics="Interaction subscription information">
           <field name="HLAinteractionClass"
              dataType="HLAhandle"
              semantics="Encoded interaction class handle"/>
           <field name="HLAactive"
              dataType="HLAboolean"
              semantics="Whether subscription is active (HLAtrue=active)"/>
        </fixedRecordData>
        <fixedRecordData name="HLAobjectClassBasedCount"
           encoding="HLAfixedRecord"
           semantics="Object class and count of associated items">
           <field name="HLAobjectClass"
              dataType="HLAhandle"
              semantics="Encoded object class handle"/>
           <field name="HLAcount"
              dataType="HLAcount"
              semantics="Number of items"/>
        </fixedRecordData>
        <fixedRecordData name="HLAinteractionCount"
           encoding="HLAfixedRecord"
           semantics="Count of interactions of a class">
           <field name="HLAinteractionClass"
              dataType="HLAhandle"
              semantics="Encoded interaction class handle"/>
           <field name="HLAinteractionCount"
               dataType="HLAcount"
               semantics="Number of interactions"/>
        </fixedRecordData>
        <fixedRecordData name="HLAsynchPointFederate"
           encoding="HLAfixedRecord"
           semantics="A particular joined federate and its synchronization point
                     status">
           <field name="HLAfederate"
              dataType="HLAhandle"
              semantics="Encoded joined federate handle"/>
           <field name="HLAfederateSynchStatus"
              dataType="HLAsynchPointStatus"
              semantics="Synchronization status of the particular joined\
                        federate"/>
```

```
         </fixedRecordData>
      </fixedRecordDataTypes>
   </dataTypes>
   <notes>
      <note
         name="MOM1"
         semantics="The value of the Dimension Upper Bound entry for the Federate
                    dimension is RTI implementation dependent."/>
   </notes>
</objectModel>
```

Annex F

(informative)

Bibliography

[B1] Naur, P. *et al*., "Report on the Algorithmic Language ALGOL 60," in *Communications of the ACM*, vol. 6, no. 1, pp. 1–17, Jan. 1963.

[B2] IEEE 100, The Authoritative Dictionary of IEEE Standards Term, Seventh Edition.